

## ThinLinc Administrator's Guide for ThinLinc 4.3.0



**Cendio<sup>®</sup>**  
**ThinLinc<sup>®</sup>**

# **ThinLinc Administrator's Guide for ThinLinc 4.3.0**

Copyright © 2014 Cendio AB

# Table of Contents

<b>I. Introduction .....</b>	<b>1</b>
1. Introduction .....	1
1.1. About the Documentation.....	1
1.2. Finding More Information.....	1
2. ThinLinc Architecture.....	3
2.1. Session Overview .....	4
<b>II. Installation.....</b>	<b>5</b>
3. Installation.....	5
3.1. Overview .....	5
3.2. Server Requirements.....	5
3.2.1. ThinLinc System and Software Requirements .....	5
3.2.2. Windows RDP Server Requirements .....	6
3.2.3. Server Sizing.....	6
3.3. Preparing the Network for ThinLinc Installation .....	7
3.3.1. A Simple ThinLinc Setup .....	7
3.3.2. ThinLinc in a Novell Network .....	8
3.3.3. ThinLinc in a Windows Network.....	9
3.3.4. ThinLinc in a NAT/Split-DNS Environment .....	10
3.3.5. Using the ThinLinc HTML5 Client .....	12
3.3.6. Other Services Required by ThinLinc Servers .....	12
3.4. Installing the ThinLinc Terminal Server .....	12
3.4.1. Starting the Installation Program .....	13
3.5. Upgrading an Old Installation .....	13
3.5.1. Acquire New Licenses .....	13
3.5.2. Run the Installation Program .....	13
3.5.3. Update Configuration Files .....	13
3.5.4. Run tl-setup.....	14
3.6. SELinux enabled distributions.....	14
3.7. The ThinLinc WTS Tools Package .....	15
3.7.1. Overview .....	15
3.7.2. Installing the WTS Tools Package on Windows Terminal Servers.....	16
3.8. VirtualGL.....	16
3.8.1. Overview .....	16
3.8.2. Installation and configuration .....	16
4. License Handling .....	19
4.1. Overview .....	19
4.2. License Counting .....	19
4.3. Location and format of License Files.....	19
4.4. Log Files and E-mail Messages.....	19
4.5. Checking the Number of Valid Licenses .....	20
5. Printer Features .....	21
5.1. Overview of ThinLinc Printer Features.....	21
5.2. Printer Configuration Overview .....	21
5.2.1. CUPS Browsing .....	22
5.2.2. CUPS configuration on the Machine Running VSM Server .....	22
5.2.3. CUPS configuration on the Machine running VSM Agent .....	22

5.3. Local printer support .....	23
5.3.1. Theory of operation.....	23
5.3.2. Device independent mode.....	23
5.3.3. Device dependent mode.....	24
5.3.4. Installation and Configuration.....	24
5.3.5. Parallel port emulation.....	24
5.4. Nearest printer support .....	25
5.4.1. Administration of the Nearest Printer Feature in ThinLinc .....	25
5.4.2. Nearest Printer Selection Algorithm .....	26
5.4.3. Printer Drivers.....	26
5.5. Printer Access Control.....	26
5.5.1. Theory of Operation.....	27
5.5.2. Requirements .....	27
5.5.3. Activating the Printer Access Control Feature .....	27
5.5.4. Configuration .....	28
5.6. Printer Configuration on Windows Terminal Servers.....	28
5.6.1. Configuration .....	29
5.6.2. Persistent Printer Settings .....	29
6. High Availability (HA).....	31
6.1. Overview .....	31
6.1.1. Background - Reasons For a HA Setup .....	31
6.1.2. Solution - Elimination of Single Point of Failure .....	32
6.1.3. Theory of Operation.....	32
6.2. Configuration of ThinLinc for HA Operations.....	33
6.2.1. Installation of a New HA Cluster.....	33
6.2.2. Reconfiguring an existing ThinLinc Installation into HA mode .....	34
6.3. Recovering from hardware failures .....	34
6.3.1. Recovering from Minor Failures.....	34
6.3.2. Recovering from Catastrophic Failure .....	35
7. The ThinLinc Client.....	37
7.1. Client usage .....	37
7.1.1. The started ThinLinc client.....	37
7.1.2. Logging in to a ThinLinc server .....	37
7.1.3. Language Settings.....	39
7.1.4. The ThinLinc session life cycle .....	39
7.1.5. The session menu .....	40
7.2. Running the ThinLinc client from the command line .....	40
7.3. Local device export .....	43
7.3.1. Sound device (Windows and UNIX only) .....	43
7.3.2. Serial ports (Windows and UNIX only).....	44
7.3.3. Drives.....	44
7.3.4. Printer.....	44
7.3.5. Smart Card Readers .....	44
7.4. Client configuration .....	45
7.4.1. Options tab .....	45
7.4.2. Local Devices tab.....	47
7.4.3. Screen tab.....	51
7.4.4. Optimization tab.....	52

7.4.5. Security tab .....	55
7.5. The XDM mode (UNIX only) .....	59
7.5.1. The XDM mode Control Panel .....	59
7.6. Logfile placement .....	62
7.6.1. UNIX log file .....	62
7.6.2. Windows log file .....	62
7.7. Client configuration storage .....	63
7.7.1. Overview and Parameters .....	63
7.7.2. Configuration Parameter Storage .....	70
7.7.3. Adding Custom Branding to the ThinLinc Client Login Window .....	70
7.8. Client Customizer .....	71
7.8.1. Introduction .....	71
7.8.2. Installation .....	71
7.8.3. Building a Customized Client .....	71
7.8.4. Adding SSH Host Keys to <code>settings.reg</code> .....	71
7.9. Advanced Topics .....	72
7.9.1. Hardware Address Reporting .....	72
7.9.2. Client Update Notifications .....	72
8. Client Platforms .....	73
8.1. Windows .....	73
8.1.1. Requirements .....	73
8.1.2. Installing the Windows Client .....	73
8.1.3. Running the Windows Client .....	73
8.2. Mac OS X .....	73
8.2.1. Requirements .....	73
8.2.2. Installing the Mac OS X Client .....	73
8.2.3. Running the Mac OS X Client .....	74
8.2.4. Command and Alt Keys on Mac OS X .....	74
8.3. Linux PC .....	74
8.3.1. Requirements .....	74
8.3.2. Installing the Linux Client .....	74
8.3.3. Running the Linux Client .....	76
8.4. Solaris .....	76
8.4.1. Requirements .....	77
8.4.2. Installing the Solaris Client .....	77
8.4.3. Running the Solaris Client .....	77
8.5. Thin Terminals .....	77
8.5.1. Neoware Terminals .....	77
8.5.2. eLux-based Thin Terminals (Fujitsu Futro et. al.) .....	77
8.5.3. VXL .....	79
8.5.4. HP ThinPro Terminals .....	79
8.5.5. IGEL Universal Desktop .....	80
8.5.6. Dell Wyse-Enhanced SuSE Linux Terminals .....	80
8.5.7. Dell Wyse-Enhanced Ubuntu Linux Terminals .....	81
8.5.8. Other Thin Terminals .....	81
8.6. Running ThinLinc on a Thinstation terminal .....	81
8.6.1. Installing and Building the Package .....	82
8.6.2. Configuring the ThinLinc client when running on a Thinstation Terminal .....	82

8.7. Web Integration and HTML5 Browser Client .....	84
8.7.1. Launching the Native Client From a Web Page .....	84
8.7.2. The CGI Script tlclient.cgi .....	85
8.7.3. ThinLinc Web Access (HTML5 Client) .....	87
9. Authentication in ThinLinc .....	91
9.1. Pluggable Authentication Modules .....	91
9.1.1. Configuration files for PAM .....	91
9.2. Limitations .....	91
9.3. Using Novell eDirectory with ThinLinc .....	91
9.3.1. Configuring eDirectory and ThinLinc with TLNC .....	92
9.3.2. Acquiring the SSL CA Certificate from eDirectory .....	95
9.3.3. Allowing Clear Text Passwords (bind operations) .....	95
9.3.4. Using eDirectory User and Group Objects with ThinLinc .....	95
9.3.5. Forcing Users to Change Passwords in an eDirectory Environment .....	99
9.3.6. Configuring Windows Terminal Servers with Netware Client for Single Sign-On	
101	
9.4. Using Public Key Authentication .....	101
9.4.1. Introduction .....	101
9.4.2. Key Generation .....	102
9.4.3. Server Configuration .....	102
9.4.4. Client Configuration .....	102
9.5. Using Smart Card Public Key Authentication .....	102
9.5.1. Introduction .....	103
9.5.2. General Requirements .....	103
9.5.3. Key Generation .....	103
9.5.4. Server Configuration .....	103
9.5.5. Client Configuration .....	104
9.5.6. Automatic Connection .....	104
9.5.7. LDAP Automatic Update (tl-ldap-certalias) .....	104
9.6. Using One Time Passwords .....	108
9.6.1. Introduction .....	108
9.6.2. General Requirements .....	108
9.6.3. Configuration for NordicEdge One Time Password Server .....	108
9.6.4. Configuration for RSA SecurID .....	109
10. File Access .....	111
10.1. Accessing Windows File Servers .....	111
10.1.1. Introduction .....	111
10.1.2. Requirements .....	111
10.1.3. Mounting and Unmounting Shares .....	112
10.2. Accessing Novell Netware File Servers .....	113
10.2.1. Introduction .....	113
10.2.2. Using NCPFS to Access Novell File Servers .....	113
10.2.3. Using the Native Novell Client to Access Novell File Servers .....	116
10.2.4. Accessing Novell Netware File Servers using NFS .....	117
10.3. Restricting write access to users home directory .....	122
10.3.1. Introduction .....	122
10.3.2. Activation .....	122
10.3.3. Configuration .....	122

10.3.4. Security Considerations and Limitations .....	123
11. Connecting to Windows Terminal Servers.....	125
11.1. Introduction .....	125
11.2. Single Sign-On .....	125
11.2.1. Information .....	125
11.2.2. Smart card .....	125
11.3. Connection Modes.....	125
11.3.1. Running a Windows Desktop in a Window .....	125
11.3.2. Running a Windows Desktop in Fullscreen.....	126
11.3.3. Running a WTS application in Standard Mode .....	126
11.3.4. Running a WTS application in SeamlessRDP Mode.....	126
<b>III. Administration .....</b>	<b>129</b>
12. Accessing Client Resources from the Terminal Server.....	129
12.1. Accessing the Clients Local Drives .....	129
12.1.1. Introduction.....	129
12.1.2. Mounting and Unmounting Local Drives .....	129
12.1.3. Accessing local drives from Windows Terminal Servers.....	130
12.1.4. Mounting Drives at Login.....	130
12.1.5. Limitations and additional information .....	130
12.2. Using Serial Port redirection .....	130
12.2.1. Introduction.....	131
12.2.2. Requirements .....	131
12.2.3. Enabling Serial Port Redirection .....	131
12.2.4. Accessing the redirected port from applications.....	131
12.2.5. Limitations and additional information .....	131
12.3. Using Sound Device Redirection .....	132
12.3.1. Introduction.....	132
12.3.2. Requirements .....	132
12.3.3. Using sound redirection with UNIX applications.....	132
12.3.4. Using sound redirection with Windows Terminal Servers.....	134
12.3.5. Limitations and additional information .....	135
12.4. Using Smart Card Redirection.....	135
12.4.1. Introduction.....	135
12.4.2. Requirements .....	135
12.4.3. Enabling Smart Card Redirection .....	135
12.4.4. Limitations and additional information .....	135
13. Commands on the ThinLinc Server .....	137
14. Server Configuration .....	147
14.1. Configuring ThinLinc Servers in a Cluster .....	147
14.1.1. Configuration Options.....	147
14.1.2. Cluster Management .....	147
14.2. Server Configuration Parameters.....	148
14.2.1. Parameters in /vsmagent/ .....	149
14.2.2. Parameters in /vsmserver/ .....	151
14.2.3. Parameters in /vsm/ .....	153
14.2.4. Parameters in /appservergroups/ .....	154
14.2.5. Parameters in /sessionstart/ .....	155

14.2.6. Parameters in /tlwebadm/.....	156
14.2.7. Parameters in /webaccess/.....	156
14.3. Configuring Logging on ThinLinc servers .....	156
14.3.1. ThinLinc server components.....	156
14.3.2. Per-Session Logging .....	158
14.4. Customizing the User's Session .....	158
14.4.1. Session startup - the big picture .....	158
14.4.2. Session startup on VSM Agent .....	160
14.4.3. Profiles and the standard xstartup.default file. ....	161
14.4.4. Session Startup with a Client Supplied Start Program .....	163
14.4.5. Configuring available profiles .....	163
14.4.6. Configuring different Linux Desktops based on the selected profile.....	165
14.4.7. Speeding up Session Startup.....	165
14.4.8. Configuring the language environment on the server based on the client language .....	165
14.4.9. Forcing sessions for some users to certain agent hosts.....	165
14.4.10. Indicating that Shadowing is in Progress.....	166
14.5. Limiting Lifetime of ThinLinc Sessions .....	166
14.6. Restricting SSH Daemon Port Forwarding .....	167
15. Hiveconf .....	169
15.1. Overview .....	169
15.1.1. Basic Syntax.....	169
15.1.2. Tree Structure.....	169
15.1.3. Mounting Datasources .....	170
15.1.4. Hostwide Configuration .....	170
15.1.5. Hiveconf Tools .....	171
15.2. Hiveconf and ThinLinc.....	171
15.2.1. The ThinLinc Configuration Tool - <b>tl-config</b> .....	171
16. Administration of ThinLinc using the Web Administration Interface .....	173
16.1. Introduction .....	173
16.2. Configuring tlwebadm .....	173
16.3. Modules .....	173
16.3.1. The System Health Module .....	174
16.3.2. The Status Module .....	174
16.3.3. The VSM Module .....	175
16.3.4. The Profiles Module.....	177
16.3.5. The Locations Module .....	179
16.3.6. The Desktop Customizer Module .....	182
16.3.7. The Application Servers Module .....	183
16.3.8. The Novell Configurator Module.....	186
17. Building Custom Linux Desktops with the ThinLinc Desktop Customizer .....	187
17.1. Introduction .....	187
17.2. Using the ThinLinc Desktop Customizer .....	187
17.2.1. Concepts.....	187
17.2.2. Using the ThinLinc Desktop Customizer .....	189
17.2.3. Handling Applications .....	189
17.2.4. Defining a Menu Structure.....	191
17.2.5. Defining Application Groups.....	192



17.2.6. Distribute Configuration to all agent hosts .....	194
17.3. Enabling the Custom Desktops for users.....	194
17.4. Tips & Tricks with TLDC .....	194
17.4.1. Unwanted Icons on the Desktop with KDE.....	195
17.4.2. File Associations for Applications Not In the Menu .....	195
17.4.3. Home Icon not Working in KDE? .....	195
<b>IV. Appendixes .....</b>	<b>197</b>
A. TCP Ports Used by ThinLinc .....	197
A.1. On Machine Running VSM Server.....	197
A.2. On Machine Running VSM Agent .....	197
A.3. On Windows Terminal Servers .....	199
B. Troubleshooting ThinLinc.....	201
B.1. General troubleshooting method.....	201
B.2. Troubleshooting Specific Problems .....	202
B.2.1. Problems Where the Client Reports an Error.....	202
B.2.2. Problems that Occur After Session Start.....	204
C. Manually Configuring Integration with Novell eDirectory .....	205
C.1. Schema extensions .....	205
C.2. Increasing performance by adding an index on some Attributes.....	205
C.3. Removing Attribute Mappings .....	206
C.4. Adding nss_map_attribute statements to /etc/ldap.conf .....	206
C.5. Creating a DN for search operations.....	206
C.6. Creating the DN used to modify users in the directory .....	208
D. Configuring CUPS queues on Windows Terminal Servers .....	209



# Chapter 1. Introduction

## 1.1. About the Documentation

This document is separated into five parts. This, the first part, is an introduction to the subject with general information about the product. The second part is about how to install different components in ThinLinc and integrate those with other systems, such as user account databases and file servers. Part three discusses the administration of ThinLinc after it is installed. The last part contains appendices with extra information.

**Note:** Before you start using ThinLinc, please read the release notes supplied in both Server and Client Bundles and online at <http://www.cendio.com/> (<http://www.cendio.com/resources/docs/relnotes/>)

## 1.2. Finding More Information

If you need more information about ThinLinc, contact your supplier and/or visit the ThinLinc homepage, <http://www.cendio.com/>. At the ThinLinc homepage you will find information about courses, upgrades, etc.

If you need more information about Linux, we recommend looking at the Linux Documentation Project homepage (<http://www.tldp.org/>) as well as the homepage for your Linux distribution.



## Chapter 2. ThinLinc Architecture

The goal of this chapter is to give a technical overview of how the system works for someone who will install or maintain a ThinLinc installation.

ThinLinc is a product for managing *server based computing*. The system is largely based on open source software, which has led to an expansion of the product to encompass solutions for authentication, availability systems, emulation and conversion between different computer systems. ThinLinc can be used as a gateway between different types of clients and a large number of base systems.

The system architecture allows an existing infrastructure to be maintained while a new architecture is gradually introduced to the organization. The system can be launched alongside the existing systems for a gradual migration to a new platform, and at the same time it acts as a link or gateway between the existing systems.

The architecture is designed to be flexible in order to handle larger organizations with autonomous office applications or functions, whilst maintaining management and security. The system can be supplemented with an automated system for installation, configuration and administration of the client hardware, such as through the use of PXE. It's also possible to create different user groups. In this way departments with special needs are easily administered in the case of adaptations or user-driven application development.

**Figure 2-1. The System Architecture of ThinLinc**

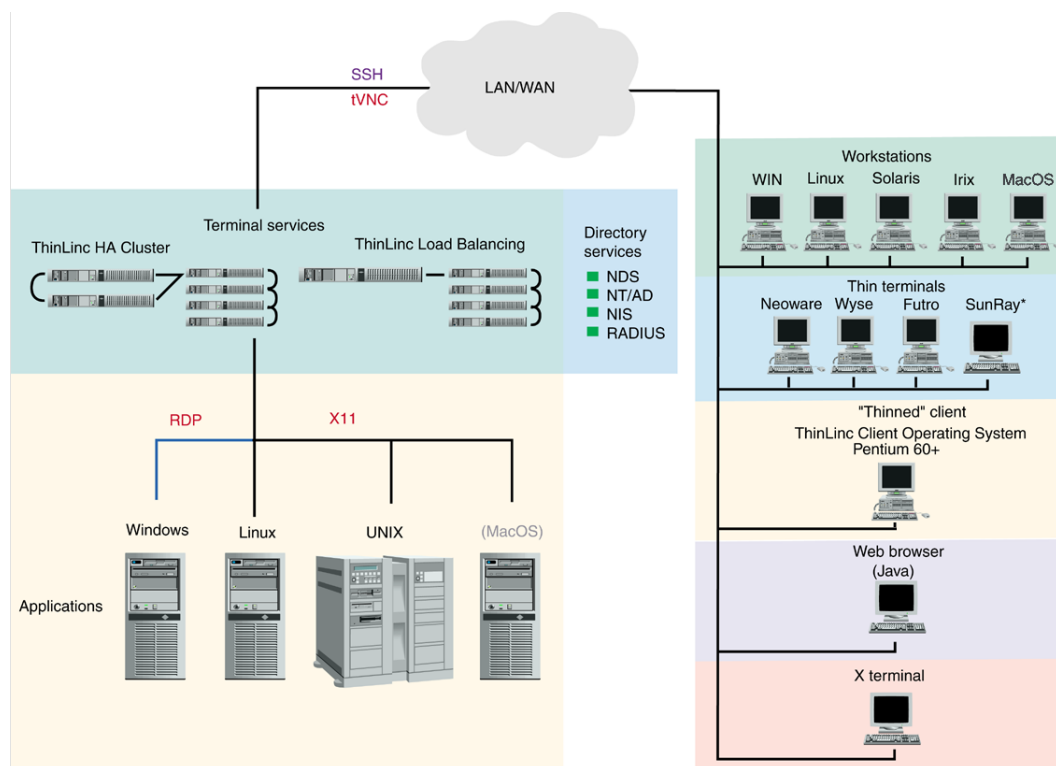


Figure 2-1 gives an overview of the ThinLinc architecture.

Several different clients can be used to connect to a ThinLinc system. Clients are available for Linux, OSX, Solaris and Windows, as well as a client for any web browser with support for HTML5.

The clients connect to a ThinLinc system located on the Local Area Network (LAN) or on a Wide Area Network (WAN) such as the Internet. Depending on the network type and the bandwidth available, several bandwidth-saving algorithms can be used to provide good performance even over narrow-banded links. Encryption is used to secure all information sent between the client and the server.

When a user connects to a ThinLinc server, a *session* is created. This session is the user's starting point for running applications either on the ThinLinc server(s) or on other servers reachable from the ThinLinc server. ThinLinc has a Single Sign-On (SSO) mechanism that enables passwordless but secure logins to (for example) Windows Terminal Servers and other Unix Servers running special applications.

The ThinLinc servers can run either Linux or Solaris. There is support for High Availability and advanced two-level load balancing.

## 2.1. Session Overview

When a user logs in from a native ThinLinc client, the following will happen:

- The client establishes a SSH tunnel to the server entered in the server field of the client interface. If this fails, then the login process will be interrupted and an error message will be displayed.
- The client tries to authenticate with the VSM server, through the SSH tunnel. The VSM server (VNC Session Manager) is the main process of ThinLinc, responsible for allocating and keeping track of user sessions.
- If the authentication succeeds, the server will check if there already exists a session for the user. If there is a session, then information about it will be returned. If there is no session a new one will be started on a terminal server and information about it will be returned. If more than one terminal server exists, load balancing will be used to select which server to start a session on.
- The client now disconnects the SSH tunnel to the VSM server and checks the information it received to see which terminal server it should connect against.
- The client now establishes a new SSH tunnel to the VSM agent server it received information about from the VSM server. Port forwarding for VNC is always established, as well as other ports depending on which local devices have been enabled. All tunnels are multiplexed over the same SSH connection.
- The client now starts the VNC viewer, which will connect to the remote VNC server via the SSH tunnel.

## Chapter 3. Installation

### 3.1. Overview

This chapter describes how to install the ThinLinc software on ThinLinc Linux Terminal Servers and MS Windows Terminal Servers. To upgrade an existing installation, see Section 3.5.

1. If your setup includes a MS Windows Terminal Server, we suggest installing this machine first. In addition, install the WTS Tools package on the Windows server, following the instructions in Section 3.7.
2. Read through any platform-specific notes for your distribution. These can be found at <http://www.cendio.com/resources/docs/platforms>.
3. Install the ThinLinc Master machine, following the instructions in Section 3.4.1.
4. Optionally, install the ThinLinc Slave machines.

### 3.2. Server Requirements

#### 3.2.1. ThinLinc System and Software Requirements

- A 32-bit, LSB-compliant Linux distribution, based on GLIBC 2.3.4 or greater, with RPM or dpkg support. An i686 (or compatible) CPU with MMX and SSE support is required.  
or  
A 64-bit, LSB-compliant Linux distribution, based on GLIBC 2.5.1 or greater, with RPM or dpkg support. An x86\_64 (or compatible) CPU is required.  
or  
Oracle Solaris® 10 on SPARC.
- GLib 2.x
- Python 2.4 or newer 2.X version
- PyGTK 2.10.0 or newer
- python-ldap (only required when using eDirectory integration, see Section 9.3)
- CUPS (Common UNIX Printing System) (only required when using nearest printer or local printers, see Chapter 5)
- An SSH (secure shell) server

As long as your platform fulfills the requirements above, ThinLinc should work as expected. As part of the quality assurance work for each release, ThinLinc is tested extensively on a few platforms. For this release of ThinLinc, the list of such platforms are:

- Red Hat® Enterprise Linux Server 7 (64-bit)

- SUSE Linux Enterprise Desktop 11 SP3® (64-bit)
- Ubuntu Desktop® 14.04 (64-bit)
- Oracle Solaris® 10 on SPARC

### 3.2.2. Windows RDP Server Requirements

- Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Both 32- and 64-bit systems are supported.

**Note:** At this time, SeamlessRDP are not supported by Windows Server 2012 and Windows Server 2012 R2.

### 3.2.3. Server Sizing

The amount of computer resources needed to run a ThinLinc cluster varies greatly with the number of users, the type of hardware used for the servers, the application mix run by the users and the type of users. Trying to estimate the number of servers needed for a specific cluster is not something that can be done using a predefined table of facts. Instead decisions should be made based on benchmarks and experience.

Below, we will try to give some ideas on what kind of resources are needed based on customer experience. With time and experience from your own cluster with your own application set, you will work out your own set of figures.

It is important to remember that the ThinLinc load balancing feature makes it easy to add another server when the need arises. Start out with a number of servers and add more as the load increases.

#### 3.2.3.1. Types of Resources

There are several types of resources needed in a ThinLinc cluster.

- *Disk*

About 100MiB of disk is needed for the software and data being part of ThinLinc. Each active session also requires a very small amount of data (normally less than 100KiB) for storage of session data and the session log. In addition to that, there must be disk available for the operating system, the applications users run and logs.

- *CPU*

The amount of CPU is very hard to estimate as it depends completely on the set of applications run by the users, and also on how active the users are as well as which response times are accepted by the users. A server that without problem copes with 100 users running LibreOffice calc updating a



spreadsheet now and then will cope with a considerably lower amount of concurrent users if they are accessing internet sites with streaming video.

When ThinLinc is used as a Windows Terminal Server frontend, meaning that the only application run is rdesktop, experience shows the amount of CPU needed is around 50-100MHz per active user.

For a full desktop (KDE or Gnome) with typical office and internet applications (LibreOffice, Firefox, some graphics program and users visiting multimedia-intensive web pages, the amount of CPU needed is somewhere between 150 and 300MHz per active user.

The CPU figures above are based on experience from customers running Intel Xeon 7140M (Netburst) CPUs. For other types of CPU, the figures should be adjusted accordingly.

- *Memory*

The amount of memory, just as the amount of CPU, is also very dependent on type application set and how active the users are.

When ThinLinc is used as a Windows Terminal Server frontend, with rdesktop being the only application run, experience shows that the amount of memory needed per user is 20-50MiB.

For a full desktop (KDE or Gnome), expect the need for 100-200MiB of memory per user, not including the memory required for individual applications.

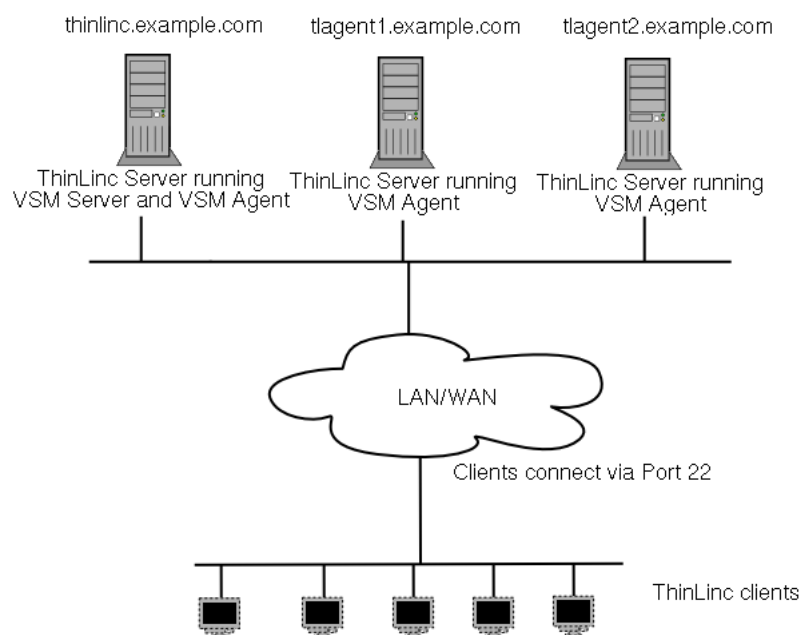
### 3.3. Preparing the Network for ThinLinc Installation

Naturally, the network at the site where ThinLinc is to be installed needs to be prepared for the installation. This section aims to help in understanding the requirements of the network for a successful ThinLinc installation.

We will explain the most common setups, including a typical Novell site and a typical Microsoft site. Also, we will explain how a site with NAT can use a NAT/Split-DNS setup to access ThinLinc in an efficient way both from the inside network as well as from the Internet.

### 3.3.1. A Simple ThinLinc Setup

**Figure 3-1. A Simple ThinLinc Setup**



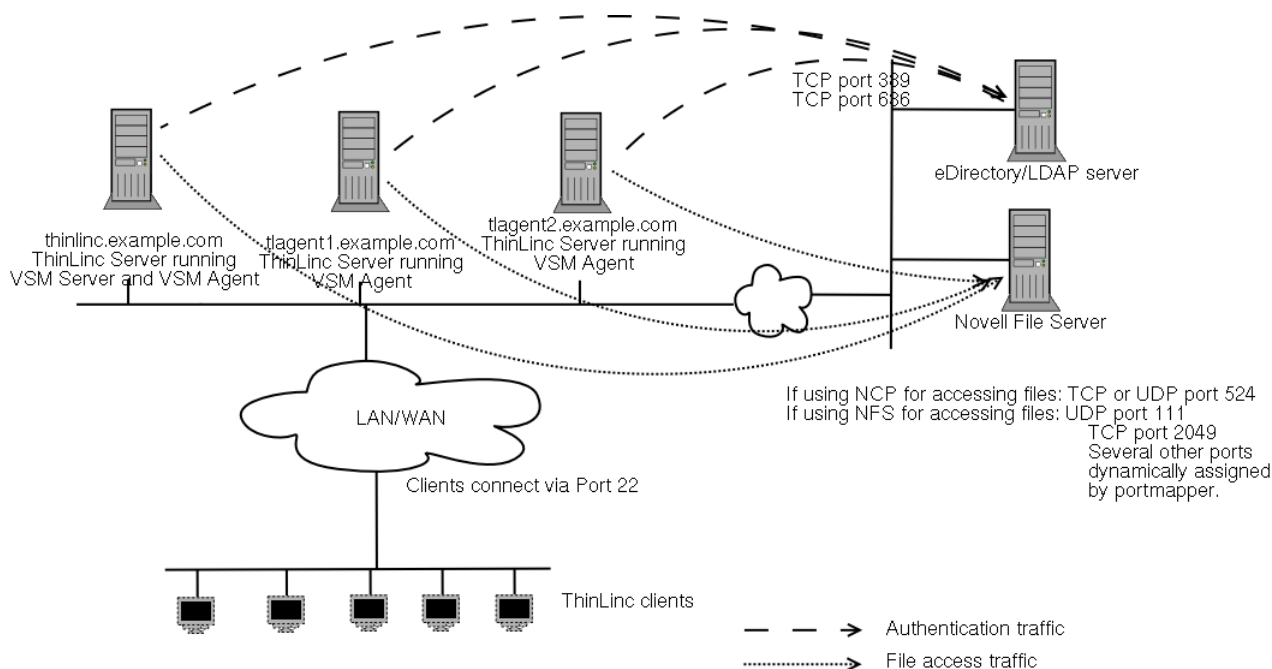
In Figure 3-1, a very simple ThinLinc setup is shown. In this setup, clients are configured to connect to *thinlinc.example.com*, DNS is configured with information about what IP addresses correspond to the hostnames *thinlinc.example.com*, *tlagent1.thinlinc.com* and *tlagent2.thinlinc.com* and no firewalls are in the path between the clients and the servers.

The number of VSM agents will range from 1 (on the same host as the VSM server) to a larger number, based on the number of users that are using the system. In this example, there are one host running both VSM server (the software controlling the whole ThinLinc cluster) and VSM agent, and two dedicated VSM agent hosts running only sessions.

Clients will communicate with the servers solely via SSH (by default port 22).

### 3.3.2. ThinLinc in a Novell Network

**Figure 3-2. ThinLinc in a Novell Network**



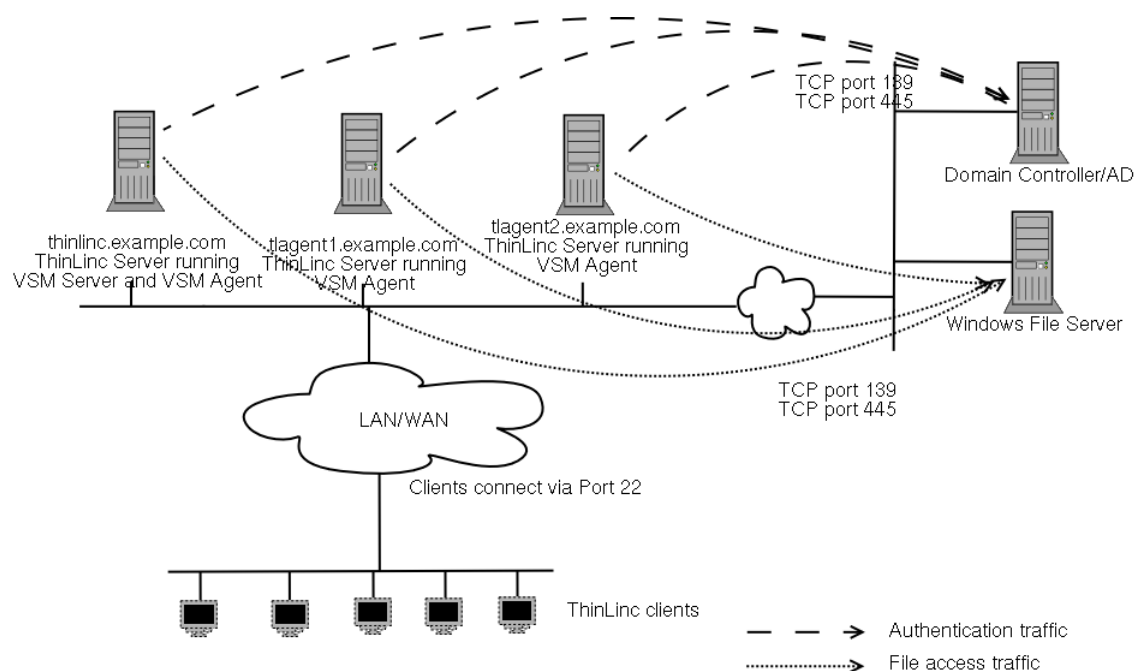
In Figure 3-2, ThinLinc is installed in a Novell environment, and integration with Novell eDirectory and/or Novell Netware filesystems are in use.

The ThinLinc servers will need to communicate with the eDirectory servers on either port 389, if using unencrypted LDAP, or on port 636, if using encrypted LDAP (ldaps).

The ThinLinc servers will also need to communicate with the Novell Netware file servers. In the case where NCP is used to access the files, the ThinLinc servers need to communicate with the Netware servers on TCP or UDP port 524. In the case where NFS is used to access files, UDP port 111, TCP and UDP port 2049 and a range of dynamically allocated UDP ports are used to communicate with the file servers. If there is a firewall between the ThinLinc servers and the Netware file servers, it needs to have support for understanding portmap requests, opening NFS UDP ports on demand, or there can be no restrictions for the traffic between the ThinLinc servers and the Netware file servers.

### 3.3.3. ThinLinc in a Windows Network

Figure 3-3. ThinLinc in a Windows Network



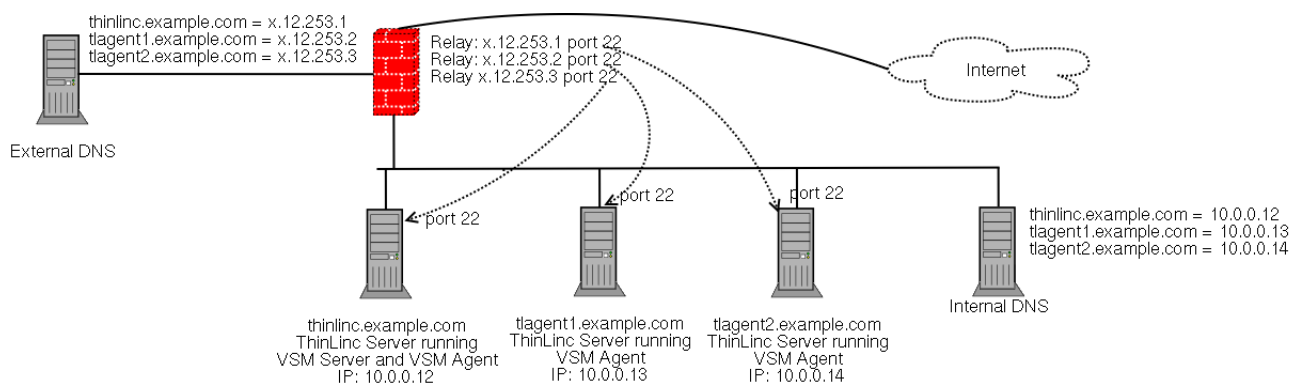
In Figure 3-3, ThinLinc is installed in a Windows environment, and integration with Windows Domain Services and/or Windows Fileservers are in use.

The ThinLinc servers need to communicate with the Windows Domain Controller on TCP port 139.

The ThinLinc servers will need to communicate with the Windows file servers using TCP port 139 and/or TCP port 445.

### 3.3.4. ThinLinc in a NAT/Split-DNS Environment

**Figure 3-4. ThinLinc in a NAT/Split-DNS Environment**



At many sites, the internal network is behind a firewall doing Network Address Translation (NAT). This means that the IP addresses on the internal network are allocated from so-called RFC1918 space, i.e., they are within the range 10.0.0.0-10.255.255.255, 172.16.0.0 - 172.31.255.255 or 192.168.0.0 - 192.168.255.255.

As long as ThinLinc servers are only meant to be accessed from the internal network, this is no problem, and the situation will be like the one described in Section 3.3.1. However, if the ThinLinc servers are meant to be accessed from the Internet as well, special arrangements need to be made.

**Note:** An alternative to using a split DNS configuration is to use a client side translation configured by the `HOST_ALIASES` parameter, but in most cases, a proper DNS setup is recommended. See Section 7.7 for more information.

#### 3.3.4.1. Relays

First, relays must be configured in the firewall. One IP address reachable from the outside network per ThinLinc server needs to be available, and each should be equipped with a relay forwarding traffic from TCP port 22 on the outside to TCP port 22 on one specific ThinLinc server. In our example, as shown in Figure 3-4, there is one relay listening to TCP port 22 on the externally reachable IP address `x.12.253.1` forwarding traffic to the ThinLinc server on the internal network with IP address `10.0.0.12`, one relay listening on TCP port 22 on the externally reachable IP address `x.12.253.2` forwarding traffic to the ThinLinc server on the internal network with IP address `10.0.0.13`, and so on.

### 3.3.4.2. DNS

After configuring the relays, DNS must be configured so DNS queries for the hostnames of the ThinLinc servers get different answers depending on the origin of the query. DNS queries originating from the internal network should be answered with the real IP addresses of the servers, and DNS queries originating from the outside network should be answered with the IP addresses on the firewall, where the relays are listening.

In our example, if a host on the internal network is asking for the IP address of the hostname *thinlinc.example.com* it should get the IP address *10.0.0.12* as answer. If a outside host is asking for the IP address of the same hostname it should instead get the IP address *x.12.253.1* as answer.

When configured this way, a client connecting from the internal network will communicate directly with the ThinLinc servers, without the need to pass the firewall, while clients connecting from the outside will pass through the firewall and the relays to communicate with the ThinLinc servers. This will ensure optimal performance for clients from the internal network, at the same time lowering the load on the firewall.

### 3.3.4.3. Configuring the VSM Agents

Finally, after configuring relays and DNS, the VSM agents must be configured to respond with the correct hostname when asked by the VSM server what hostname the clients should connect to. The default behaviour is to respond with the IP adress of the host, but that will not work in this case since clients connecting from the external network won't have any route to for example *10.0.0.13*. Instead, the VSM agents should be configured to respond with the hostnames that can be found in both the internal and the external DNS.

This is done by setting the parameter `/vsmagent/agent_hostname` on each of the VSM agents in the ThinLinc cluster. In our example, set `/vsmagent/agent_hostname` to *tlagent1.example.com* on the machine with IP adress *10.0.0.13*.

### 3.3.5. Using the ThinLinc HTML5 Client

If users are supposed to be able to connect using a web browser, using the ThinLinc HTML5 Browser Client, they must be able to connect to port 300 on both the VSM server and on all VSM agents.

In the NAT/Split-DNS setup, relays must obviously be configured in the firewall for each ThinLinc server and the port 300.

### 3.3.6. Other Services Required by ThinLinc Servers

In order for ThinLinc to function properly together with the rest of the network, they will need to synchronize time with some internal or external time source. Linux machines use the Network Time Protocol (NTP), so if there is one or several NTP servers on the internal network, the ThinLinc servers will need to communicate with them. Otherwise, the ThinLinc servers should be configured to use some external time source, and should be allowed to communicate with it.

## 3.4. Installing the ThinLinc Terminal Server

### 3.4.1. Starting the Installation Program

The installation program is located in the root directory of the Server Bundle. Extract the bundle and start the installation program as follows:

```
sh ./install-server
```

If you prefer, you can also install the ThinLinc packages by hand. These packages are located in platform-specific subdirectories in the Server Bundle; either `serverkit-linux` or `serverkit-solaris`.

After installing the software packages, Thinlinc must be configured. This is done by the program `/opt/thinlinc/sbin/tl-setup`. If you are running `install-server`, it will ask you if you want to start `tl-setup` at the end of the package installation.

## 3.5. Upgrading an Old Installation

Upgrading an old installation of ThinLinc is very much like installing it from scratch. The only difference is that you will have to adapt the old settings in `/opt/thinlinc/etc/conf.d/` to the new configuration files afterwards.

### 3.5.1. Acquire New Licenses

Before performing an upgrade, find out if you need new license files to run the new version. ThinLinc license files delivered with version *x.y.z* will still work for versions with the same *x* and *y* but higher *z*, but not for increased *x* or *y*. For example, license files for ThinLinc 3.1.0 will still work for ThinLinc 3.1.1, but not for ThinLinc 3.2.0 or ThinLinc 4.0.0.

Contact your reseller for new licenses. If you bought ThinLinc with a maintenance agreement, new licenses will be provided without cost.

As the new licenses will work with the old (current) version, it's a good idea to install them as the first step in the upgrade process.

### 3.5.2. Run the Installation Program

The same installation program that you used to install ThinLinc is also used to upgrade it. It is located in the root directory of the Server Bundle. Extract the bundle and start the installation program as follows:

```
sh ./install-server
```

and answer the questions. If you prefer, you can also upgrade the ThinLinc packages by hand. These packages are located in platform-specific subdirectories on the Server Bundle; either `serverkit-linux` or `serverkit-solaris`.

### 3.5.3. Update Configuration Files

When upgrading ThinLinc, the package installation process handles configuration files that have been changed by taking backups of existing configuration. The installation program will prompt the user to adjust them before running **tl-setup**. See below for instructions on how to handle the configuration files on different types of systems.

#### 3.5.3.1. RPM-based Linux systems

Look for files with filenames ending in `.rpmsave`, `.rpmorig` in `/opt/thinlinc/etc` and below. These are copies of the files as they were before the upgrade. Review the differences between the old and the new files, and add relevant statements from the old files to the new files, then remove or move away the `.rpmsave` and/or `.rpmorig` files.

#### 3.5.3.2. DPKG-based Linux systems

Look for files with filenames ending in `.dpkg-old` in `/opt/thinlinc/etc` and below. These are copies of the files as they were before the upgrade. Review the differences between the old and the new files, and add relevant statements from the old files to the new files, then remove or move away the `.dpkg-old` files.

#### 3.5.3.3. Solaris

All configuration files that were present in the old version which you are upgrading from will be saved as `<filename>.pkgsave`. Transfer relevant statements from these files to the new configuration files installed by **pkgadd**.

### 3.5.4. Run tl-setup

After installation of the packages and modification of the configuration files as described above, run `/opt/thinlinc/sbin/tl-setup` to verify that the system is correctly configured.

## 3.6. SELinux enabled distributions

ThinLinc is designed to run with reference SELinux policy and users in the unconfined context. It is possible to use ThinLinc with other policies and more restricted contexts, but will most likely require modifications to your policy to accommodate ThinLinc.

The local system policy will optionally be modified by **tl-setup** during installation. The SELinux module and other policy changes performed can be examined in `/opt/thinlinc/share/selinux`. Execute the command `/opt/thinlinc/share/selinux/install` to reapply ThinLinc's policy changes.

**Note:** The ThinLinc policy module is distributed in source form and therefore requires the reference policy build environment. On Red Hat based systems this is always installed, but other systems might require extra packages.



## 3.7. The ThinLinc WTS Tools Package

### 3.7.1. Overview

The ThinLinc WTS Tools package contains support software for Microsoft Windows Terminal Servers. This includes:

#### tl-loadagent

ThinLinc has a feature where sessions against Windows Terminal Servers are distributed among several available hosts. In order for this to work, the `tl-loadagent` service must run on all Windows Terminal Servers.

For information about which ports are used when communicating with the load balance agent, refer to Appendix A.

#### tl-is-appsession

The `tl-is-appsession` utility allows you to detect if the WTS session is running a full desktop, or just an application. This is done by examining the RDP Startup Shell. When a desktop session is detected, this command returns 1. Otherwise, 0 is returned. This utility is useful in login scripts. For example, it might be desirable to open up a browser whenever a new desktop session starts. This can be done with a script like this:

```
%ProgramFiles%\ThinLinc\WTSTools\tl-is-appsession
if %errorlevel% == 1 start http://intranet
```

#### The SeamlessRDP Shell

The SeamlessRDP Shell is the server component required for SeamlessRDP.

#### ThinLinc WTS sound driver

The ThinLinc WTS sound driver `tlsnd` replaces the native `rdpsnd` sound driver normally included with Microsoft Terminal Services. This driver is needed to get sound capture (microphone) support.

**Note:** The ThinLinc WTS sound driver is currently only supported on Windows 2003, Windows 2003 R2, and Windows XP Professional.

#### ThinLinc GINA

The ThinLinc GINA extends the Microsoft GINA by adding support for smart card single sign-on. This means that smart card authenticated connections to Terminal Services from a ThinLinc session can be initiated without entering the PIN code again. This requires that the "Send smart card passphrase (PIN) to server" client option is enabled. See Section 7.4.5 for more information.

**Note:** The ThinLinc GINA is currently only relevant in Active Directory configurations. When using Novell eDirectory, use the Novell GINA instead.

**Note:** The ThinLinc GINA is not supported on Windows 7, Windows Server 2008 or later.

### 3.7.2. Installing the WTS Tools Package on Windows Terminal Servers

Installation of the WTS Tools package is easy. Simply execute the **tl-wts-tools.exe** program from the `windows-tools\wts-tools` directory in the Server Bundle, and answer the questions.

To activate the ThinLinc WTS sound driver, follow the instructions below:

- Disable the built-in sound redirection. On Windows Server 2003, this can be done using the Terminal Services Configuration tool. The checkbox "Audio mapping" on the Client Settings tab should be checked.
- Import the registry file "Activate ThinLinc sound driver" that can be found under "ThinLinc WTS Tools" in your start menu.
- The ThinLinc WTS driver only works in conjunction with the PulseAudio (<http://pulseaudio.org/>) sound system. Therefore, make sure that PulseAudio and the application `padsp` is installed on all ThinLinc servers. The application server group must also be configured to use PulseAudio. See Section 14.2.4 for more information.

To activate the ThinLinc GINA, select the icon "Activate ThinLinc GINA". A reboot is recommended.

Before you uninstall the WTS Tools Package it is crucial that you deactivate both the ThinLinc sound driver and the ThinLinc GINA, by using the icons found under "ThinLinc WTS Tools" in your start menu.

## 3.8. VirtualGL

### 3.8.1. Overview

VirtualGL is used to provide server-side hardware 3D acceleration to applications displayed on a remote client. VirtualGL can be used with ThinLinc to provide accelerated graphics for OpenGL applications running in Linux environment.

Although ThinLinc is designed to work in combination with VirtualGL, VirtualGL is not developed or maintained directly by Cendio AB, and as such is not shipped as a part of the ThinLinc product.

### 3.8.2. Installation and configuration

Full documentation regarding the installation and configuration of VirtualGL can be found online at <http://www.virtualgl.org/Documentation/Documentation>.

**Note:** The following section numbers references the VirtualGL 2.3.3 documentation. Documentation for past or future VirtualGL releases may have different section numbers.

For the general case, it should be sufficient to consult the following sections:

- 5.1 - Installing VirtualGL on Linux
- 6.1 - Granting Access to the 3D X Server

And see also:

- 9.1 - Using VirtualGL with an X Proxy on the Same Server

For more advanced configuration, such as using a remote application server with VGL Transport, see the following sections:

- 6.3 - SSH Server Configuration
- 8 - Using VirtualGL with the VGL Transport

**Note:** Publishing applications in this way is not supported by default in ThinLinc, for example by using `tl-run-unixapp`. Applications published in this manner will need to be called from a script using `vglconnect`, and likely some form of non-interactive authentication, e.g. public key. This script may then be made available to users by specifying it as an application within TLDC - see Chapter 17.



## Chapter 4. License Handling

### 4.1. Overview

To run a session against a ThinLinc cluster, the server must be equipped with license files. The license files specify the number of concurrent users the cluster is allowed to run.

If no license files are installed on the cluster, a maximum of ten concurrent users are allowed.

Each cluster can have one or several license files. Each file contains licenses for a specific number of concurrent users. When the VSM Server starts up, it reads all license files and creates a sum of the number of concurrent users allowed based on the licenses from all files.

License files have one soft and one hard limit. When the soft limit is reached, new sessions can still be started, but a license violation will be logged and sent to the administrator (see Section 4.4). If however the hard limit has been reached, new sessions cannot be started. The purpose of this system is to allow growing organisations some time to adapt the number of licenses to a growing number of concurrent sessions, avoiding loss of production.

### 4.2. License Counting

One license is required for each pair of (*username, client hardware*). This means that if a user runs several sessions from the same client, only one license is used. If the same user runs multiple concurrent sessions from different client hardware, multiple licenses are required by the user.

### 4.3. Location and format of License Files

License files are delivered either in the form of text files (filename extension `.license`) or ZIP files (filename extension `.zip`). Transfer each file to your ThinLinc cluster and place it in `/opt/thinlinc/etc/licenses`. Make sure that the transfer of the files uses binary mode, or the license file might not be verifiable. We recommend transferring via `scp` or `sftp`.

After adding new license files, either restart VSM Server by running `/opt/thinlinc/libexec/service vsmserver restart` or wait until the VSM Server automatically reads in the new licenses, something that happens once every 12 hours.

**Note:** When running VSM Server in a High Availability setup (see Chapter 6), license files should be copied to `/opt/thinlinc/etc/licenses` on both nodes.

### 4.4. Log Files and E-mail Messages

ThinLinc logs user license violations to the file `/var/log/thinlinc-user-licenses`. Other license-related messages are logged to `/var/log/vsmserver.log`.

If license violations occurs, ThinLinc sends email to the person defined as system administrator in the parameter `/vsmserver/admin_email` in `vsmserver.hconf`. E-mail messages warning about license violations are sent every 12 hours if any license violations have occurred.

## **4.5. Checking the Number of Valid Licenses**

You can use the program `/opt/thinlinc/sbin/tl-show-licenses` to verify the number of valid user licenses. There is also a graph available in the administrative interface. See Chapter 16 for more information.

## Chapter 5. Printer Features

### 5.1. Overview of ThinLinc Printer Features

ThinLinc has several printer-related features that aims to provide the user with maximum flexibility while making the administrator's work easier. A ThinLinc system normally uses CUPS (Common Unix Printing System) to provide normal printing services. By integrating with CUPS, ThinLinc also provides the following features:

- *Local Printer support* allows users to print documents on a printer that is connected to their terminal from applications running on the ThinLinc server.

See Section 5.3 for documentation on this feature.

- *Nearest Printer* is a feature that simplifies the printing process for the user by automatically printing to a printer that is located at the terminal the user is currently using. Users only need to know that they should always print to the *nearest* printer - the system will figure out the rest based on a database of terminals, printers and locations, eliminating the need to learn the names of printers at different locations. This decreases the need for support.

See Section 5.4 for documentation on this feature.

- *Printer Access Control* uses the same database of terminals, locations and printers as the *Nearest Printer* feature to dynamically limit which printers a user may print to based on the terminal the user is currently using. This feature also limits the list of printers seen by each user to the printers the user are allowed to use, simplifying choice of printer for the user by only showing the printers that are relevant at the current location.

See Section 5.5 for documentation on this feature.

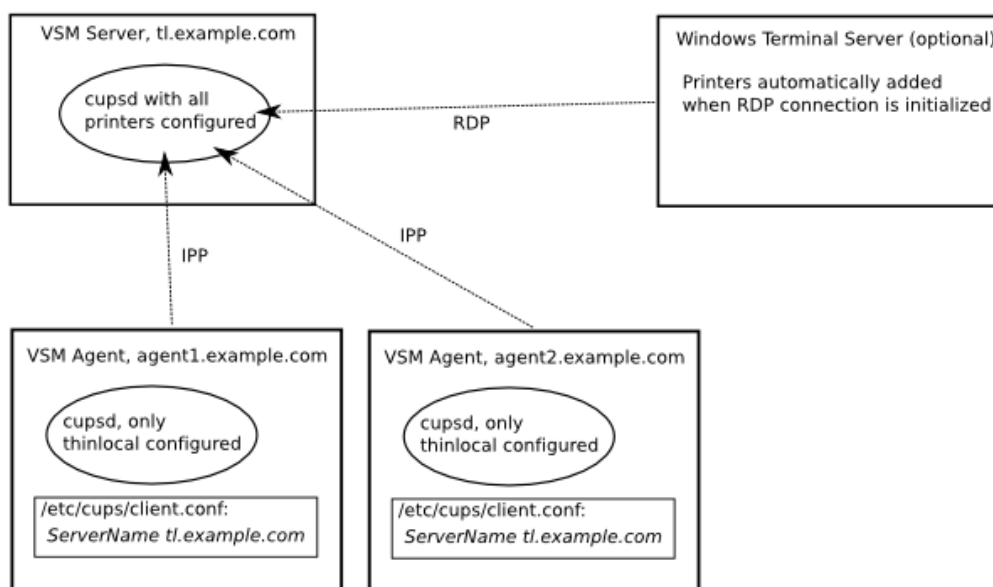
- *Printing from Windows Terminal Servers* is handled by automatic redirection via RDP. All printers the user has access to in his/her Linux environment are automatically added to the WTS session.

See Section 5.6 for documentation on this feature.

### 5.2. Printer Configuration Overview

This section provides an overview of how printing is configured in a ThinLinc cluster.

**Figure 5-1. Printer Configuration Overview**



### 5.2.1. CUPS Browsing

It is important that the CUPS Browsing feature is turned *off* on all machines in the cluster, or problems with duplicate thinlocal printers will occur.

### 5.2.2. CUPS configuration on the Machine Running VSM Server

Configure all printers that need to be available in the CUPS configuration on the machine running VSM Server. Either use distribution-specific tools, or the built-in administration interface in CUPS which can usually be reached by using a web browser, connecting to port 631 on the machine, i.e. <http://tl.example.com:631/>.

The nearest and thinlocal queues, used by the nearest printer and the local printer features respectively, are added by **tl-setup** when installing ThinLinc.

Printers, with one exception (see below) only needs to be configured on the machine running VSM Server. Agent nodes will use the CUPS daemon (cupsd) on the VSM Server machine for printing.



### 5.2.3. CUPS configuration on the Machine running VSM Agent

The machines in the cluster that run VSM Agent, i.e., the machines that host user sessions, need a running CUPS daemon (cupsd), but this cupsd only needs one printer defined - the *thinlocal* queue. The reason for this is that the local printer backend needs to run on the same machine as the session of the user printing to local printer to be able to access the endpoint of the SSH tunnel used to transport the printer job to the client.

The thinlocal queue is added by **tl-setup** when installing the agent.

**Note:** The CUPS daemon on each agent must listen to requests on the network interface, and allow printer jobs from the machine running VSM Server to be submitted to the thinlocal queue.

When a user submits a job to the local printer, i.e. to the thinlocal queue, the printer job will be submitted to the CUPS daemon running on the VSM Server host. It will then be respooled to the cupsd on the agent server hosting the session. This is to make central configuration of all other printers possible.

## 5.3. Local printer support

### 5.3.1. Theory of operation

With ThinLinc, it is possible to print to a printer attached to the client computer. Two primary modes of operation are available: device independent and device dependent. Both modes can be used at the same time. See below for details about the two modes.

The thinlocal printer is cluster-aware. If a user submits a print job on a node in a ThinLinc cluster which does not host the user's session, the print job will automatically be respooled to the correct node. This is used in the recommended setup (see Section 5.2).

If a user has more than one session, print jobs submitted to the local printer will be redirected to the client that made the last connection.

The local printer features are implemented as a backend to CUPS (Common Unix Printing System).

**Note:** When using local printers, we recommend that you activate the parameter `/vsmserver/unbind_ports_at_login`.

### 5.3.2. Device independent mode

The device independent mode is designed to provide universal access to any local printer without having to install drivers on the ThinLinc server. This is achieved by converting the print job to the Adobe Portable Document Format (PDF) on the terminal server, and then sending it through an encrypted tunnel to the client. The client subsequently prints the job on the local printer using a built-in PDF renderer.

Because the driver on the ThinLinc server is device independent, it has no way to know what capabilities (duplex ability, trays, paper size, etc.) the printer connected to the client has. At the same time, applications that want to print need to know about these capabilities to print correctly.

As a compromise, the universal printer is configured with a PPD (Postscript Printer Definition) that covers a broad range of printer capabilities - it's a *Generic Postscript Printer* driver. This makes it possible for CUPS to convert input formats to the correct format before sending them to the local printer. It also means that default values can be set for some of the configuration parameters, for example paper size, using the CUPS configuration interface.

### 5.3.3. Device dependent mode

The device dependent mode is to be used when it is necessary to access all options on the printer, or when the communication with the printer cannot be expressed in terms of normal pages (e.g. a label printer). In this mode the printer driver is installed on the ThinLinc server and the data is sent unmodified to the local printer.

**Note:** ThinLinc has no way of verifying that the connected printer is the correct one, so it is up to the user to make sure that a device dependent queue is not used with a different printer.

### 5.3.4. Installation and Configuration

Use **tl-setup** to install the PDF conversion filter, the backend and queue in CUPS on all machines running VSM Agent. This adds a new queue named *thinlocal* to CUPS and makes it available to your users. This queue is the one to use for device independent mode described above.

After installation, the local printer is ready for use. Make sure your ThinLinc client is configured to allow redirection of printers, then print to the *thinlocal* queue, and the job will be rerouted to the default printer of the client you're currently using.

Device dependent queues are installed as if installing the printer locally on the ThinLinc server. The only difference is that the URI shall be specified as `thinlocal:/.` Example:

```
# lpadmin -p thinlocal-label -v 'thinlocal:/' -P /media/cd/label-printer.ppd
```

### 5.3.5. Parallel port emulation

ThinLinc also includes a very basic form of parallel port emulation that gives legacy application access to the local printer. It is built on top of the *thinlocal* queue, which means it only works if certain requirements are satisfied:

- The application must only write to the port. Reading is not supported, neither is monitoring or altering the port status pins.

- After a print job is completed, the application must close the port. As the emulation is unaware of the printer protocol, closing the port is the only way it can determine where one job ends and another begins.

To access the emulated parallel port, configure the application to use the port

```
$TLSESSIONDATA/dev/lp0.
```

## 5.4. Nearest printer support

With the ThinLinc nearest printer feature, printer jobs are routed to a printer near the terminal the user is currently sitting at. This is accomplished by matching printers to hardware addresses of terminals.

The *nearest printer* is implemented as an extra printer queue, above the real printers. Printer jobs sent to the nearest queue will be sent to the nearest printer backend. The backend is a program which is called by CUPS together with all needed information. The nearest backend will look at the user name handled by CUPS and then ask the ThinLinc VSM server for more information about this user. The information tells the backend which terminal the user is currently using. It then queries the information stored in Hiveconf for a list of printers close to the terminal used by the printing user. When a printer is known the backend will reprint the job to this printer.

This queue is added to the VSM master server by **tl-setup** at installation of ThinLinc. The recommended setup is to configure one nearest printer queue in the CUPS daemon on the VSM Server host, and then let all agents use this CUPS daemon. See Section 5.2 for an overview of printer setup in a ThinLinc cluster.

### 5.4.1. Administration of the Nearest Printer Feature in ThinLinc

To be able to work properly the nearest printer system needs information about physical layout. The information is divided into three sections, printers, locations and terminals. This information can be administrated using the ThinLinc Web Administration.

Each printer that should be handled by the nearest printer system should be entered with a name and an optional comment. The button **Fetch printers from CUPS** can be used to fetch the current list of printers from CUPS. Please note that printers that are removed from CUPS must be manually removed from the list of printers in tlwebadm.

Each location with terminals should be entered with a name, an optional comment and a list of local printers. A location can for example be a classroom, a department, a house, and so on. The possible printers are the ones entered in the printer interface.

For each terminal entered in this system you enter the terminal network interface hardware (MAC) address and the location the terminal resides in. The hardware address can be entered in many formats, but will be converted to all uppercase hexadecimal form separated by colon, i.e. "01:23:45:67:89:AB". You can explicitly add printers to the terminal settings, which then gets higher priority than the printers defined at the terminal's location. This can be handy if a terminal has a local printer connected to it, perhaps in a large room where the other printers are far away.

Normally you will first enter all printers, then all locations and finally all terminals in the system. To each location you can add a list of printers near that location. If the location is so big that different printers are close to different parts of the location, then you should probably divide the location into smaller parts. You can also assign close printers to a terminal. This can be used in cases where a terminal

has a printer attached to itself or if the terminal, part of a bigger location, is placed in its own room together with a printer. You can read more about this administration in Chapter 16.

### 5.4.2. Nearest Printer Selection Algorithm

If a terminal has a printer directly assigned to it in the terminals module in `tlwebadm`, that printer will be the nearest printer for that terminal. For printers without a printer directly assigned (the normal situation), the first printer in the list of printers for the terminal's location is selected when the user submits a printer job to the *nearest* queue.

If a user has more than one session, print jobs submitted via nearest printer will be redirected to the printer that is most near the client that made the last connection.

### 5.4.3. Printer Drivers

When printing via the nearest printer, the CUPS client can't get hold of all information about the real printer where the job will actually be printed, because it doesn't know that the printer job will be rerouted by the nearest driver. Therefore, the printing application has no way to know about the number of trays, the paper sizes available etc.). This is a problem for some applications, and it also adds to the number of applications that will be misconfigured, for example selecting the wrong paper size.

As a compromise, the nearest printer is configured with a PPD (Postscript Printer Definition) that covers a broad range of printer capabilities - it's a *Generic Postscript Printer* driver. This makes it possible to configure default values for some of the settings, for example paper size, using the CUPS configuration interface.

If all the printers in your organisation are of the same type, it may be a good idea to replace the Generic Postscript PPD installed for the nearest queue with a PPD for the specific printer in use. That will let CUPS-aware applications select between the specific set of features available for the specific printer model.

## 5.5. Printer Access Control

In a ThinLinc cluster, all printers that any user of the cluster needs to be able to print to must be defined centrally, or the user will not be able to print from applications that run in a ThinLinc session. For large installations, this leads to a very long list of available printers.

A long list of printers leads to usability problems - having to select printer from a long list can be troublesome. Also, it opens for problems with printer jobs being printed at remote locations by mistake (or on purpose, by users finding it amusing to send "messages" to other locations).

The solution to this problem is the Printer Access Control feature of ThinLinc. By integrating with CUPS (the Common Unix Printing System), the list of printers a user is presented with and allowed to print to is limited to the printers that should be available to a specific terminal, based on information in a database of printers, terminals and locations.

**Note:** The Printer Access Control feature will affect all users on the ThinLinc cluster. The only user excepted from limitations of the printer list is the superuser (root) - all other users will only see and

be able to use printers based on the location of their terminals, when the Printer Access Control feature is enabled.

### 5.5.1. Theory of Operation

Each time a user requests a new session or reconnects to an existing session, the hardware (MAC) address of the terminal is sent along with the request from the ThinLinc client. Using the same database as the *nearest printer* feature used to find which printer is closest to the user, the printer access control feature calculates which printers the user is allowed to use, and then configures the access control of the printing system (CUPS).

This way, the user is presented with a list of printers that only contains the printers relevant for the location where the terminal the user is currently using is located. In a situation where a user has multiple sessions running from multiple clients, all printers associated with the different terminals will be made available.

### 5.5.2. Requirements

- CUPS v1.2 or higher.

### 5.5.3. Activating the Printer Access Control Feature

First, make sure you have configured the printers in your ThinLinc cluster as documented in Section 5.2. For the Printer Access Control Feature, a central CUPS daemon on the VSM Server host is required, and all agent hosts must have a correctly configured `/etc/cups/client.conf`.

To activate the printer access feature, create two symlinks on the host running VSM Server, as follows:

```
ln -s /opt/thinlinc/sbin/tl-limit-printers /opt/thinlinc/etc/sessionstartup.d
ln -s /opt/thinlinc/sbin/tl-limit-printers /opt/thinlinc/etc/sessionreconnect.d
```

The first symlink makes sure **tl-limit-printers** is run when sessions are started. The second makes sure it is run at reconnects to existing sessions. More details about the session startup can be found in Section 14.4.

**Note:** With the above configuration (symlinking `tl-limit-printers` into `sessionstartup.d` and `sessionreconnect.d`), the client will not get an answer back from the server until `tl-limit-printers` has finished its execution. This is the desired behaviour if it is strictly necessary that printer access rights are correct when the user connects to the session. In environments where it is acceptable that the final list of printers shown to the user may not be finished when the user connects to the session, place the execution of `tl-limit-printers` in the background, as detailed in Section 14.4.1.1, as that will decrease the time the user has to wait for the session to appear on his client.

After creating the symlinks, try connecting to your ThinLinc cluster with a ThinLinc client and bring up an application that lists the available printers. The list of printers should now be limited according to configuration.

**Note:** The printer list limitation doesn't work for applications that use the deprecated *cupsGetPrinters* library call. This means that older applications might show the whole list of printers. The access control is still enforced, which means that even if a disallowed printer is shown in the list of printers, users can't submit jobs to it.

Most applications in a modern Linux distribution doesn't have this problem.

### 5.5.4. Configuration

Configuration of the printer access control feature is mostly a matter of using *tlwebadm* (see Chapter 16 for details) to add the hardware address of all terminals as well as information about where they are located and which printers are to be available for each location.

#### 5.5.4.1. Unknown Terminals / Terminals Without Hardware Address

When a client reports a hardware address that is not present in the database of terminals, or when no hardware address is reported, the default behaviour is to disallow access to all printers, rendering an empty printer list for the user.

There is however a way to give even unknown terminals access to one or more printers - define a special location and check the *Use for unknown terminals and terminals without hardware address* checkbox. Then add the printers that should be available for the unknown terminals.

One common configuration is to add such a location and then add the *thinlocal* printer to this location. This way, unknown terminals, for example people working from their home computers, will be able to use their locally connected printer, but no other printer will be available.

## 5.6. Printer Configuration on Windows Terminal Servers

If your ThinLinc setup uses a Windows Terminal Server for some applications, access to printers for these applications is handled automatically by redirection over RDP. When *rdesktop* is started via **tl-run-rdesktop** as is the case when applications on Windows Terminal Servers are started via any of the ThinLinc wrapper commands, the list of printers available to the user on the Linux or Solaris server running the ThinLinc session is automatically fetched. Each printer is then added to the *rdesktop* commandline in a way that makes it appear in the user's session on the WTS.

One advantage of this feature is that the Section 5.5 will be active even for programs running on Windows Terminal Servers, since only the printers the user are allowed to see and use are exported.

For some special cases, manually configuring printers on the Windows Terminal Server may be required. This is the case for example if some application requires that the name of the printer is exactly the same across sessions, as printers added automatically have names that contain the WTS session ID. For information on how to configure printers manually, see Appendix D

The same printer driver is used on the Windows Terminal Servers for all printers regardless of model. This limits the amount of settings a user can modify when printing. For example, selecting a different paper tray than the default one is not possible. If these limitations are a problem, defining printer queues manually as described above may be an option.

### 5.6.1. Configuration

The automatic redirection of printers to Windows Terminal Servers is enabled by default, but can be disabled by setting `/appservergroups/rdp/<appgroup>/redirect_printers` to *false*.

### 5.6.2. Persistent Printer Settings

If a user modifies the printer settings for a redirected printer on a Windows Terminal Server, the changes are written to a file named `~/.rdesktop/rdpdr/<prntername>/AutoPrinterCacheData`.

If changes to printer settings should be made for all printers, login to the WTS, make the changes, then wait a few minutes, log out, then copy the generated

```
~/.rdesktop/rdpdr/<prntername>/AutoPrinterCacheData to
/opt/thinlinc/etc/rdpdr/<prntername>.
```

This must be done for all printers.

The settings file will now be copied to user home directories when **tl-run-rdesktop** is run. Existing files will not be overwritten, so if printer settings need to be changed, existing files in user home directories must be removed.





## Chapter 6. High Availability (HA)

### 6.1. Overview

This chapter describes how to setup ThinLinc with High Availability (from now on referred to as "HA") for the VSM server, providing protection against the single point of failure that the hardware running the VSM server normally is.

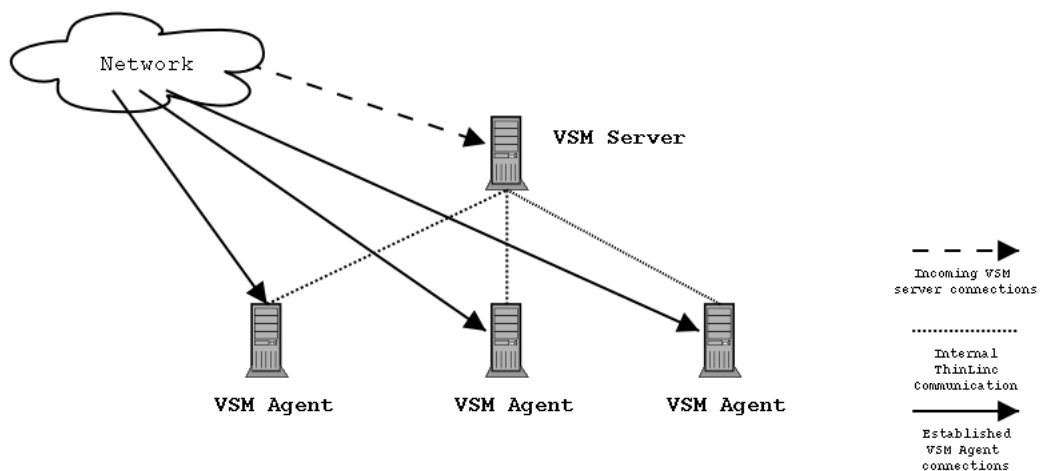
The basic principle behind this setup is to have two equal machines, both capable of running VSM server. If one of the machines goes down for some reason, the other machine will take over and serve VSM server requests with no or short interruption of service.

**Note:** The HA functionality provided by ThinLinc provides synchronization of the ThinLinc session database across two VSM servers. The software used by these machines to implement failover is not part of ThinLinc, and must be installed and configured according to your requirements. The industry standard for doing so on Linux is provided by the Linux-HA project; see <http://linux-ha.org> for more information.

#### 6.1.1. Background - Reasons For a HA Setup

In a standard ThinLinc setup, there is a single point of failure - the machine running the VSM server. If the VSM server is down, no new ThinLinc connections can be made, and reconnections to existing sessions can't be established. Existing connections to VSM agent machines still running will however continue to work. A ThinLinc cluster of medium size with one machine running as VSM server and three VSM agent machines is illustrated in Figure 6-1

Figure 6-1. A non-HA ThinLinc cluster setup



Here the incoming connections are handled by the VSM server which distributes the connections to the three VSM agent machines. If the VSM server goes down, no new connections can occur. The VSM server is a single point of failure.

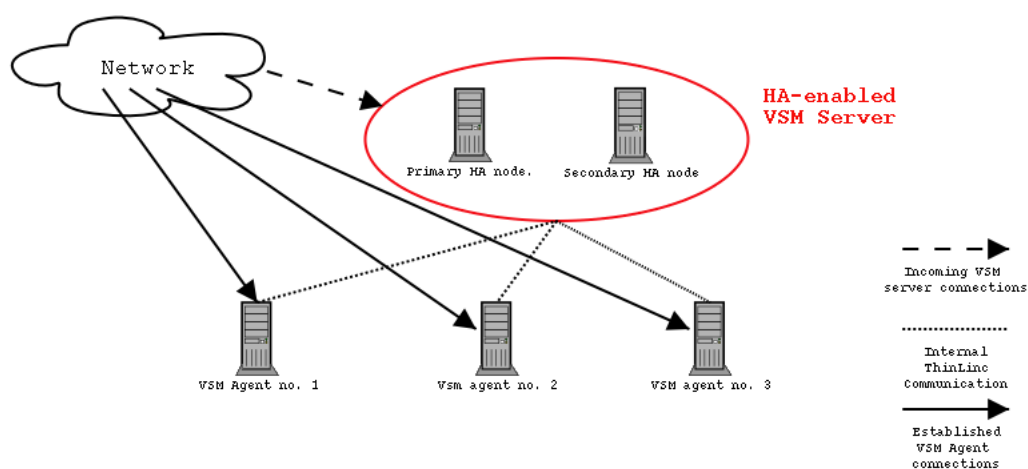
### 6.1.2. Solution - Elimination of Single Point of Failure

In order to eliminate the single point of failure, we configure the VSM server in a HA configuration where two machines share the responsibility for keeping the service running.

The two machines are in constant contact with each other, each checking if the other one is up and running. If one of the machines goes down for some reason, for example hardware failure, the other machine detects the failure and automatically takes over the service with only a short interruption for the users. No action is needed from the system administrator.

### 6.1.3. Theory of Operation

Figure 6-2. A ThinLinc HA cluster setup



In a HA setup, as illustrated in Figure 6-2 two equal machines are used to keep the VSM server running. One of the machines is primary, the other one is secondary. The primary machine is normally handling VSM server requests, but if it fails, the secondary machine kicks in. When the primary machine comes online again, it takes over again. That is, in normal operation, it's always the primary machine that's working, the secondary is just standby, receiving information from the primary about new and deleted sessions, maintaining its own copy of the session database.

Both machines have an unique hostname and an unique IP address, but there is also a third IP address that is active only on the node currently responsible for the VSM server service. This is the so called HA address, the address the clients are connecting to.

## 6.2. Configuration of ThinLinc for HA Operations

In this section, we describe how ThinLinc is configured for High Availability.

### 6.2.1. Installation of a New HA Cluster

In this section, we will describe how to setup a new HA cluster. In the examples we will use a primary node with the hostname *tlha-primary* and IP address 10.0.0.2, a secondary node with the hostname *tlha-secondary* and IP address 10.0.0.3, and a HA IP address of 10.0.0.4 with the DNS name *tlha*.

1. Begin by installing ThinLinc as described in Chapter 3 on both nodes.
2. Both nodes in the HA cluster must have the same SSH host key. Copy `/etc/ssh/ssh_host_*` from the primary host to the secondary host, and restart ssh on the secondary host.
3. Install and configure the system-level high-availability software, for example the software provided by the Linux-HA project, which can be found at <http://linux-ha.org>. This and other high-availability software may also be provided as part of your distribution, so check for the solution which best fits your requirements before proceeding.
4. Configure the system's high-availability software to watch the status of the other machine via the network, and to enable the HA IP address *10.0.0.4* on the active node. The machine with the hostname *tlha-primary* should normally be active.
5. Configure each VSM agent to allow privileged operations both from *tlha-primary* and *tlha-secondary*:

```
[root@agent root] tl-config '/vsmagent/allowed_clients=tlha-primary tlha-secondary'
```

Also, set the master\_hostname to the DNS name of the HA interface:

```
[root@agent root] tl-config /vsmagent/master_hostname=tlha
```

Restart all VSM agents after changing the configuration values.

If the `tl-config` command is not found, logout and login again in order to let the login scripts add `/opt/thinlinc/bin` and `/opt/thinlinc/sbin` to the PATH.

6. Verify operations of VSM Server on both nodes. Make sure you can start the VSM server properly on both hosts, and connect to the respective hosts when VSM server is running (i.e., it should be possible to connect, using `tlclient`, to both *tlha-primary* and to *tlha-secondary*).

Both nodes should be configured with the whole list of VSM agents in `/vsmserver/terminalservers`.

### Warning

It is **VERY IMPORTANT** that `127.0.0.1` is not in the list of terminal servers. If the machines running VSM server are also VSM agents, their unique hostnames or IP addresses must be added to the `/vsmserver/terminalservers` instead of `127.0.0.1`. The reason for this is that `127.0.0.1` will be a different server based on which VSM server is currently active.

7. After verifying that normal ThinLinc connections work as intended when using both the primary and the secondary VSM server's hostname, it is time to enable HA in the VSM servers. This is done by setting `/vsmserver/HA/enabled` to 1, and by specifying the nodes in the cluster in `/vsmserver/HA/nodes`. For example:

```
[root@tlha-primary root] tl-config /vsmserver/HA/enabled=1
[root@tlha-primary root] tl-config '/vsmserver/HA/nodes=tlha-primary.example.com tlha-secondary.example.com'
```

Configuration should be identical on both nodes. Restart the VSM server on both nodes after configuration.

8. If `vsmserver` can't safely determine which of the two nodes in `/vsmserver/HA/nodes` is the remote node, and which is the local node, it will start without HA enabled, and log a message. If this happens, validate your hostname and DNS setup. One of the entries of `/vsmserver/HA/nodes` must match the local machine. Either the resolved IP of one of the entries in `/vsmserver/HA/nodes` must match the local IP, or one entry must exactly match the local hostname as returned by `uname -n`.
9. Once HA has been configured, tests should be performed in order to confirm that the failover works as expected. This can normally be done by simply removing the network cable from the primary node, and ensuring that the secondary node then takes over. Check also that any active ThinLinc sessions have been synchronized from the primary to the secondary node, and that logging in to such a session on the secondary node succeeds once the primary node has been disabled.

Your ThinLinc HA cluster is now configured! When sessions are created, changed or deleted on the currently active node, the information about them will be transferred to the other node using a inter-VSM server protocol. If the other node has to take over service, its copy of the session data will be up to date, and it can start serving new requests instantly. When the primary node comes up again, the secondary node will resynchronise with the master.

### 6.2.2. Reconfiguring an existing ThinLinc Installation into HA mode

If you have an existing ThinLinc installation and want to eliminate the single point of failure (the VSM server), the procedure is very much like the procedure for installing a new HA cluster.

## 6.3. Recovering from hardware failures

If situations occur where the secondary node has been forced to take over service because the primary node failed for some reason, it's important to know how to recover.

### 6.3.1. Recovering from Minor Failures

If the primary went down because of a minor failure (overheating trouble, faulty processor, faulty memory etc.) and the contents of the files in `/var/lib/vsm` are untouched, recovery is very simple and fully automatic. Simply start the server and let the two VSM servers resynchronize with each other.

### 6.3.2. Recovering from Catastrophic Failure

If a catastrophic failure has occurred, and no data on the disks of the primary can be recovered, ThinLinc needs to be reinstalled and HA must be reinitialized.

Install ThinLinc as described in Section 6.2.1, but before starting the VSM server after enabling HA in the configuration file, copy the file `/var/lib/vsm/sessions` from the secondary to the primary. That will preload the database of active sessions with more current values on the primary.



## Chapter 7. The ThinLinc Client

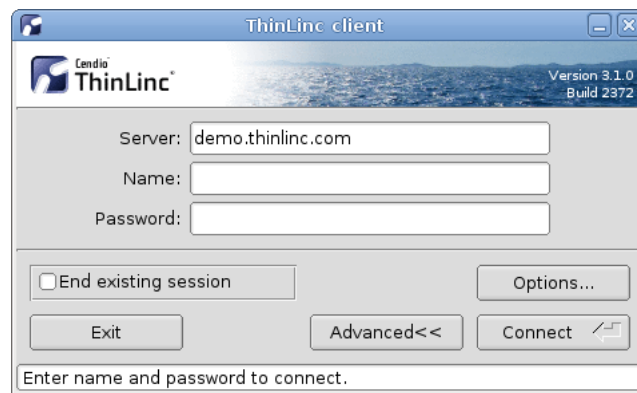
### 7.1. Client usage

Starting the ThinLinc client is normally easy, but the method can differ somewhat between the available operating systems. See Chapter 8 for instructions on how to start the client on different platforms.

#### 7.1.1. The started ThinLinc client

When the ThinLinc client is started it will show the login window. This window contains a ThinLinc logo, text fields where needed information can be entered, buttons for control and at the very bottom a status field that gives information about the login procedure.

**Figure 7-1. The ThinLinc client login window**



#### 7.1.2. Logging in to a ThinLinc server

To login into a ThinLinc server the client needs to do a successful user authentication. This means that it needs to tell the ThinLinc server a user name and a corresponding authentication information (a password or an encryption key). The ThinLinc server controls that the information is valid and accepts or denies the login attempt.

The things the client needs to know to successfully login the user against a ThinLinc server is a server address, a user name and the corresponding authentication information. When the client is normally started it will display two text fields labeled "Server" and "Name", and one text field labeled "Password", "Key" or "Certificate". This can differ some depending on command line arguments, but this is the normal behavior.

Accepted values for the server field is the hostname or the IP address of the server. The name field should be filled in with the ThinLinc username. The authentication information needed depends on the type of authentication used:

- For password authentication, a plain text password should be entered. The password won't be shown as clear text when entered.
- For public key authentication, the path to an encryption key must be entered or browsed to using the "..." button.
- For smart card authentication, a certificate must be chosen using the drop down menu next to the certificate name field.

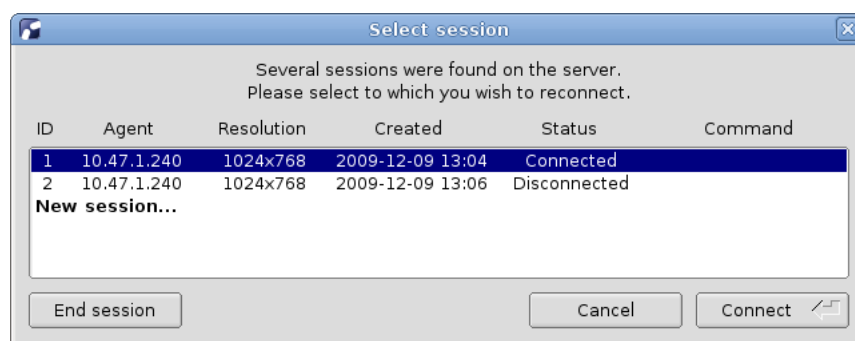
The server name, username, key path and certificate name are saved when the user tries to start the session, so they don't have to be entered again each time a new session is wanted.

When the user has entered server address, username and authentication information, it becomes possible to login. This is done by pressing the *Connect* button or the *enter key* on the keyboard. The client will then try to establish a connection with the ThinLinc server. If any of the fields has a bad value that prevents the client from successfully logging in, for example if the username or password is incorrect, there will be response message shown as a message box with the suiting information.

**Note:** By default, usernames are case-sensitive when logging in via the ThinLinc client. This behaviour may be changed using an option in the client configuration file - see `LOWERCASE_LOGIN_NAME` in Section 7.7 for details.

If the login attempt is successful a ThinLinc session will start, an old one will be reused or a session selection box might be presented, all depending on the client's settings and how many sessions the user has running, a . See Section 7.4.1 for more information on how the choice is made.

**Figure 7-2. The ThinLinc client session selection window**





The session selection window presents the user with a list of relevant sessions and several buttons to act on those sessions:

#### Connect

Connect to the selected session, or create a new session if the current selection is "Create new session...".

#### End session

Forcefully terminate the selected session and restart the connection procedure.

#### Cancel

Abort the connection and return to the main window.

The server will then prepare a graphical session on a ThinLinc server. The client then connects to this session and displays it. Normally the user now sees a dialog with different session options. The user can there select for example to run a Linux session or a Windows session. Depending on the choice the server at the other end will start that kind of session.

### 7.1.3. Language Settings

The ThinLinc client gets all its strings from a database. This way it can be easily translated, by just providing a new database for a new language.

On Unix based systems, the client picks up which language to use by reading the standard POSIX *locale* environment variables. A somewhat simplified description of these follow here:

- `LC_ALL` : If this environment variable is set, it takes precedence over all other locale variables. It will affect all locale settings, including message strings, sorting order, money representation, decimal numbers, etc.
- `LC_MESSAGES` : If `LC_ALL` is not set but this one is, it will make the messages of the client adhere to the language in question, in effect making the client use that language. There are several other variables of this kind, but they do not affect the ThinLinc client.
- `LANG` : If `LC_ALL` is not set then the value of this variable will be used for all locale categories that are not explicitly set, e.g. `LC_MESSAGES` .

There is also a variable called `LANGUAGE` on some systems, but it is non-standard, and we do not recommend the use of it.

If none of these variables are set, the locale defaults to C, which in practice means American English. The value of the variables should be of the form *language\_country*, where language and country are 2 letter codes. Currently, the languages delivered with the client are Brazilian Portuguese (`pt_BR`), English (`en_US`), Dutch (`nl_NL`), French (`fr_FR`), German (`de_DE`), Italian (`it_IT`), Russian (`ru_RU`), Spanish (`es_ES`), Swedish (`sv_SE`), and Turkish (`tr_TR`).

On Windows, the same environment variables can be set in a script that also starts the ThinLinc client. An example script called `altlang.cmd` is installed with the ThinLinc client for Windows. If nothing is set, the Windows client will use the language setting that was given with the control panel.

### 7.1.4. The ThinLinc session life cycle

When the user has started a ThinLinc session the client login interface disappears from the desktop. The client program continues to run in the background as long as the ThinLinc session is running. The client enters a service mode where it handles services needed to fulfill the requested features. For example the client handles the export of local printers, serial ports, and so on. When the ThinLinc session quits the client service engine quits as well.

There are several ways a session can end. The most common one is that the user chooses to logout from the session. That causes the session to finish on the server side. The ThinLinc server finds out that the session has finished and disconnects the client. Another possibility is to intentionally disconnect the client, without finishing the session on the server. This can be done by using the session menu. See Section 7.1.5 below for information about how to do this. When the client disconnects before the session running on the server is told to end, then the session will continue to run on the server. The next time the user logs in the server will reconnect the user to the very same session. This way it's possible to, for example, disconnect a session at work, go home, reconnect to that session and continue to work.

If the user knows that there already is a session running on the server, but still wants to start a new fresh session, then it's possible to check the *End existing session* check box that exists in the client login interface (advanced mode only). The client will then tell the server that it wants to end the existing session (if it exists) and get a new one.

### 7.1.5. The session menu

When the ThinLinc session is authenticated and the ThinLinc session is running it's possible to control the session. For example it's possible to change between full screen mode and window mode, and to disconnect the ThinLinc client from the server.

To switch to windowed mode there is a session menu that pops up when the user press a predefined key. The default key for this is F8, but the key is configurable from the client options. See Section 7.4.1 for more information about how to change this key. In the session menu you should select *Full screen* to toggle full screen mode.

## 7.2. Running the ThinLinc client from the command line

To run the ThinLinc client from the command line you run the program `tlclient`, optionally followed by options and a server name. The correct program syntax is as follows.

```
tlclient [options] [server][:port]
```

The optional *server* field can be used to specify a ThinLinc server that should be predefined in the server field when client is started. The optional *port* parameter causes the client to try to connect another TCP/IP port number than the normal SSH port when establishing it's secure connection to the ThinLinc server. More information about custom SSH settings is available at Section 7.4.5.

The ThinLinc client is highly controllable from the command line by the use of command line arguments. Many parts of the client can be controlled this way. The more simple things to control is the server or user name. It is possible to force settings and lock tabs and fields in the config interface to prevent them from being changed.

All arguments written on the command line overrides the settings saved from previous sessions. The options window will show the current settings, including the settings from the command line. The client settings is only stored to file when the user press the *OK button* in the options window. This means that options from the command line normally don't affect the saved settings. But if the user opens the options window and accepts the settings by pressing the *OK button* then the settings, including the one from the command line, will be saved.

For a complete list of arguments supported by your client you can run the client with the argument `-?`.

## Description of available command line arguments

Here follows a description for all available command line arguments.

`-?, --help`

Display a help summary.

`--version`

Display client version information and exit.

`-a, --advanced`

Start client in advanced mode. Advanced mode means that the client will show the *Server field*, *Options...* button and the *End existing session* checkbox. The advanced mode is the normal mode used when you start the ThinLinc client. A simpler mode, where those interface components are hidden, is used automatically when you enter a server name as a command line argument. By adding this argument you override that and always use the advanced mode.

`-c, --controlpanel`

Enable the Control Panel and apply stored mouse and keyboard speed and acceleration settings if any. Only available on UNIX versions of the client.

`-C, --configfile FILE`

Specifies an additional configuration file. Parameter values in this configuration file overrides the values specified by the system wide and user configuration file. Settings changed from the GUI will be stored in this configuration file, instead of the user's configuration file.

`-d, --debug LEVEL`

The ThinLinc client logs information about the current session to the file `~/.thinlinc/tlclient.log` on UNIX systems and `%TMP%\tlclient.log` on Windows systems. When the client is started, any existing log file is renamed as `tlclient.old.log`. The amount of information to log can be configured with this option followed by a number from 1 to 5. A low number gives less logging than a higher number. The default is a log level of 3. For more information about log file placement, see Section 7.6 below.

`-u, --user USER`

This option sets the user name that should be filled in into the *Name* field. This can be used to override the name that is automatically saved from last session. If you for example, in a school classroom, want it to always start with an empty Name field, then you can use this parameter with the empty string `""`.

**-p, --password PASSWORD**

This option sets the password that should be filled in into the *Password* field. When this option is used and a user name exists (either saved from previous session or entered with the *-u* parameter) the client will automatically try to login, directly after start. If the login try fails it will return focus to the client interface, making it possible to adjust the values. Note that the command line of *tlclient*, and therefore the password, will be visible to other processes running on the client operating system. If this is a problem in your environment, consider using the *-P* option documented below.

**-P, --askpass PROGRAM**

This option makes it possible to specify an askpass program that should be used to achieve the password. This program should in some way ask the user for a password and then return that password together with an exit code. This triggers the auto login (see argument *-p* above).

**-e, --encodings ENCODING, ...**

This option makes it possible to select which VNC encoding you want to use (see Section 7.4.4 for more information about VNC encodings). Valid encodings for this option are: *Tight*, *ZRLE*, *Hexile* and *Raw*.

**-l, --lock ITEM, ...**

This option makes it possible to lock different parts of the client interface. This can be used to prevent things from being changed. Locked parts will still be shown, but will be "grayed out", which means that they can't be made active for change. The items that should be locked should follow this option as a comma separated list. The following items are possible to lock.

- *server*: server entry field
- *thinlinc*: ThinLinc tab
- *options*: options tab
- *screen*: screen tab
- *optimization*: optimization tab
- *ssh*: SSH tab

**-h, --hide ITEM, ...**

This option makes it possible to hide different parts of the client interface. This can be used to remove parts of the interface that can confuse novice users, or to prevent them from reaching parts of the interface. The following items are possible to hide.

- *options*: options button
- *controlpanel*: control panel button (only in XDM mode)

**-f, --force SETTING, ...**

This option makes it possible to force a setting to a value. This can be used to preset a client with values and to force them to reset to those values each time, even if the users make changes. When an option is forced it is turned on. The following items are possible to force.

- *terminate*: terminate session
- *fullscreen*: fullscreen mode
- *sound*: sound mode
- *sshcomp*: ssh compression

**-M, --minimize**

This option causes all other applications to be minimized when the ThinLinc client starts.

**-s, --startprogram**

Specifies the program to start in the session. Overrides the `START_PROGRAM_ENABLED` and `START_PROGRAM_COMMAND` configuration parameters.

**--loop**

This option causes the client to run forever. The exit button is removed, and when a session has ended, a new client process is automatically started.

**Note:** The only way to stop the client from restarting is to terminate the `tlclient` process.

## 7.3. Local device export

ThinLinc supports export of different local devices. This means that a device that exists on your client computer or terminal can be reached from the ThinLinc session that runs on the server. The type of devices that can be exported varies depending on which operating system the ThinLinc client runs on. The export is, very generalized, done by establishing secure tunnels for the data transmission and services that connect both ends. Here follows more information about each type of possible export; for detailed information about how to enable each type of export in the client, see Section 7.4.2 below.

### 7.3.1. Sound device (Windows and UNIX only)

This feature makes it possible to hear sound from applications that runs on the ThinLinc server. Sound will be sent from the ThinLinc server to your local client through a secure connection. A small local sound daemon will be automatically started by the ThinLinc client. A secure tunnel for sound will be established during the ThinLinc session setup.

All programs that support the Enlightened Sound Daemon (EsounD) or PulseAudio should automatically be aware of this tunnel and send their sound to the client. See also Section 12.3 for information about supporting other applications.

The sound data that is sent from the server session to the local client is uncompressed audio data. This means that it can be relatively large and may use relatively much network bandwidth. This feature should not be used if you plan to use ThinLinc over low bandwidth connections such as modems or ISDN connections.

### 7.3.2. Serial ports (Windows and UNIX only)

This feature makes it possible to export two local serial ports to the ThinLinc session. When serial port redirection is enabled, a small redirection daemon will be automatically started by the ThinLinc client during session startup. A secure tunnel for serial port data will be established.

#### Warning

When activating serial port redirection, all users on the terminal server can access the serial port of the client machine.

### 7.3.3. Drives

This feature makes it possible to, in a secure way, export one or many local drives from the client machine to the server session. This can be local hard disk volumes, local CD-ROM drives, and so on. The local drive will be made available on the ThinLinc server session.

Each exported device can have individual permission settings. All export settings are made in the ThinLinc client options interface.

### 7.3.4. Printer

This feature makes it possible to export a local printer to make it available from the ThinLinc session. When enabled, the client will setup a secure tunnel for printer jobs. The client will also activate a small built-in print server that listens for printer jobs on this tunnel.

When you print to the special printer queue *thinlocal* in your ThinLinc session, then the job will be sent through this tunnel and then printed on the client machine. On UNIX platforms, the print job will always be sent to the default printer. On Windows and Mac OS X, it is possible to select whether the print job should be sent to the default printer or if the printer selection dialog should be used every print. Note that device dependent print jobs will always go to the default printer.

For more information about printer redirection in ThinLinc, see Section 5.3.

### 7.3.5. Smart Card Readers

This feature makes it possible to export all local smart cards and smart card readers to make them available from the ThinLinc session. All smart card readers available to the system will be exported to

the session so there is nothing to configure except an activation switch.

The ThinLinc client relies on the PC/SC interface present on the system to communicate with the smart card readers. If you have a reader that uses another system, then that reader will not be exported.

**Note:** Smart card export on Mac requires OS X version 10.5.6 or later.

## 7.4. Client configuration

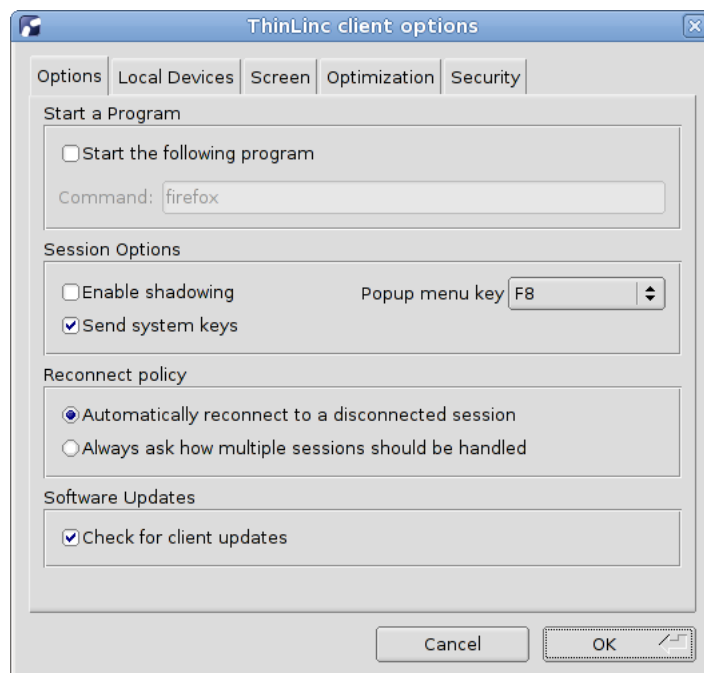
To configure the ThinLinc client you press the button labeled *"Options..."* in the client window. That brings up the client options window. This window contains several pages of settings, ordered in tab sets. The following sections will describe each of these pages and all individual settings.

When a user press the *OK button* all the current settings in the options window is saved. For more information about the config file format, see Section 7.7.

### 7.4.1. Options tab

The Options tab contains general options for the ThinLinc session. This includes settings for which program to execute in the session, shadowing another users session, reassignment of session pop-up key and how reconnections are handled.

**Figure 7-3. Client settings Options tab**



## Description of options tab settings

Here follows detailed description of the settings available in the options tab.

### Start a Program

If enabled, the client requests that the server should start the session with the command supplied by the client. Otherwise, the session command is determined by the server configuration.

### Enable shadowing

When enabled, client shadowing will be enabled in the client. An extra text field will be present in the client main window. This field is used to enter the user name of the user whose session you want to shadow.

### Send system keys

When this setting is enabled and the client is in full screen mode, key combinations such Alt+Tab will be sent to the remote system instead of being handled locally. To regain access to the local system without ending the session, the menu key must be used.

### Popup menu key

During a ThinLinc session you can press a specific key to bring up the session control pop-up window. This window can for example be used to toggle to and from full screen mode and to disconnect the session. The default key for this is *F8*, but other keys can be configured here. The feature can also be disabled by selecting *None*.

### Reconnect policy

When the client connects to a ThinLinc server, there might already be multiple sessions running on it. Some of these sessions might be connected to another client, and some might be disconnected. The client can be configured to automatically handle some of these cases, or always ask the user what to do.

**Note:** Sessions that have been started with a command different from the one currently used will be ignored.

### Automatically reconnect to a disconnected session

1. If there is no disconnected session and additional sessions are allowed, a new session is created.
2. If there is a single disconnected session then that will be used.
3. If server allows only one session, reconnect to existing session.



4. Otherwise, ask how to proceed.

### Always ask how multiple sessions should be handled

1. If there is no running session, then create a new one.
2. If server allows only one session, reconnect to existing session.
3. Otherwise, ask how to proceed.

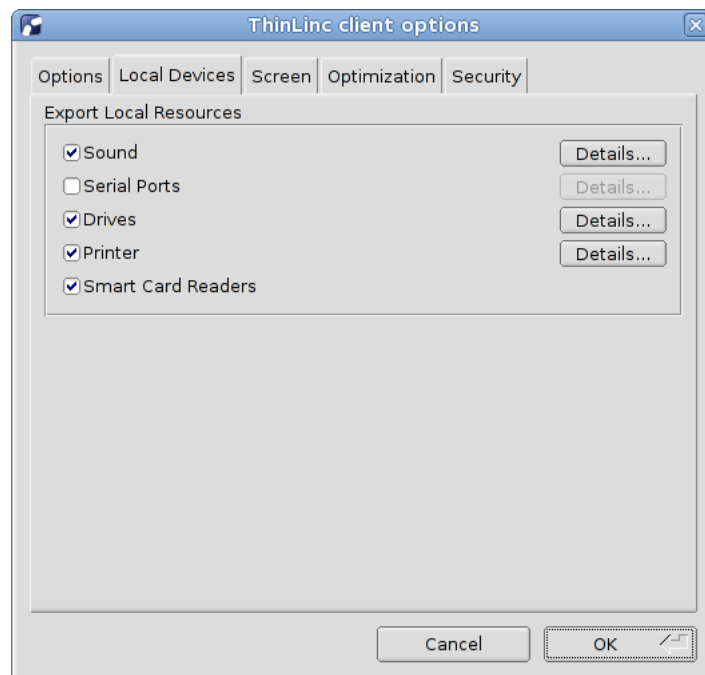
### Software Updates

If enabled, the client will periodically query the `UPDATE_URL` value specified in `tlclient.conf` for updates. If a newer version is available, the user will be asked if they want to install it.

## 7.4.2. Local Devices tab

The Local Devices tab contains options for which local devices should be exported to the server and in what manner.

**Figure 7-4. Client settings Local Devices tab**



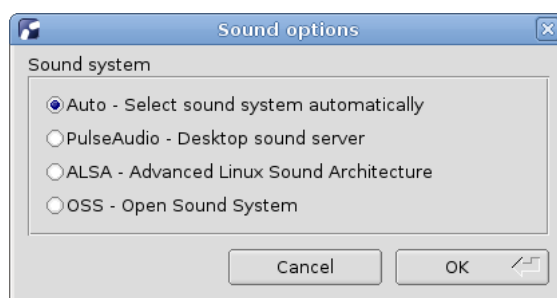
## Description of local devices tab settings

Here follows detailed description of the settings available in the local devices tab.

### Export - Sound Device

When enabled, sound will be sent from the ThinLinc server to your local client. A small local sound daemon will be started by the client, which connects to a secure tunnel to the server. See Section 12.3 for more information about this topic.

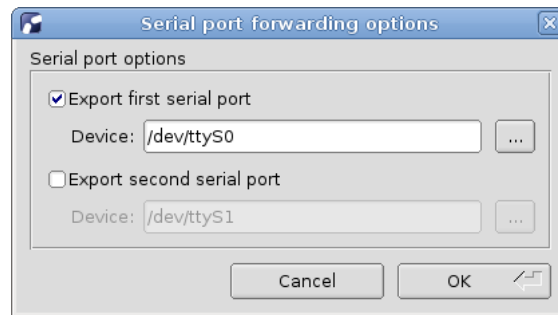
**Figure 7-5. Sound system selection interface**



On Linux there is a "Details..." button next to the Sound check box that will allow you to choose between PulseAudio, ALSA and OSS for the local sound system. You can also let the ThinLinc client select the correct system automatically.

### Export - Serial Ports

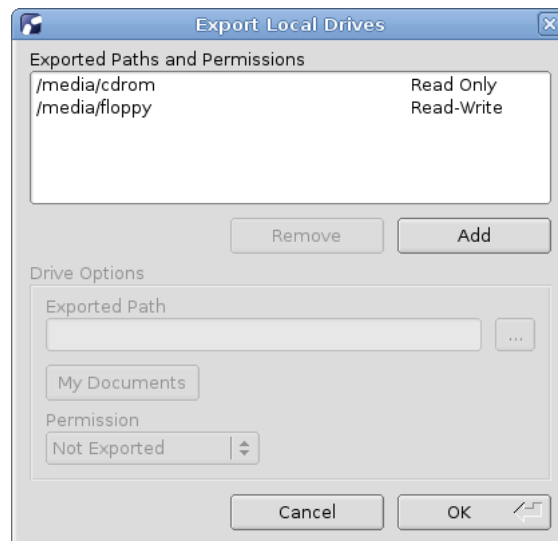
It is possible to forward two serial ports from the client to make it available to programs you run on the server. To select which of your local serial devices to export you can press the "Details..." button next to the Serial Port check box. This will bring up a dialog where you can select which two serial ports should be exported.

**Figure 7-6. Serial port selection interface**

The *Device* should be a path to a UNIX serial device (such as `/dev/ttyS0`) or a Windows COM port name (such as `COM1`). Enter the device to export in the text field or press the "..." button to browse to the wanted device.

### Export - Drives

This check box turns on export of local devices from your terminal top the ThinLinc server. This makes your local drives available from your ThinLinc session. To select which drives to export you press the "Details..." button next to Drives check box. That presents a dialog where you can build a list of drives to export and set export permissions.

**Figure 7-7. Local drive export selection interface**

The *Export Local Drives* window consists of two parts. At the top there is a list containing exported paths, with two control buttons below. The lower half contains settings fields for the currently selected path. When you select a path listed in the upper list you will see its corresponding settings in the Drive Options field below. You can then change the selected path by changing the values on the options field.

To add a new path to the list you press the *Add* button. That creates a new empty land in the path list. The new path will be automatically selected. you can then modify the settings in the lower half. Set the path and export permission for the new export. To set the export path you can either write it manually in the path text field or press the "... " button to bring up a file navigation window.

To remove a path you simply select a path and press the *Remove* button.

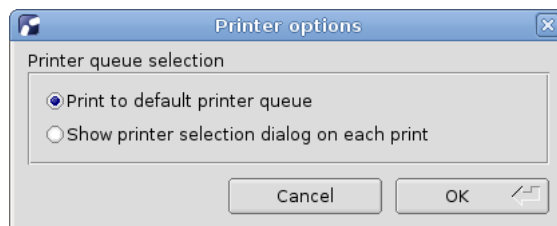
The Windows client features a mechanism that makes it easy to export the "My Documents" folder. This feature is activated by pressing the "My Documents" button. Regardless of the local folder name, this folder will be mounted as "MyDocuments" on the server.

The export permissions can be one of the following three options, *Not Exported*, *Read Only* and *Read-Write*. The *Not Exported* option can be used to temporarily turn off an export without having to delete it. The *Read Only* option means that you from the ThinLinc session will be able to read from the export, but not write. The *Read-Write* option means that you from the ThinLinc session will be able to both read and write.

### Export - Printer

By checking this check box the client will export your local printer to make it available from the ThinLinc session. For more information about this feature, see Section 7.3.4 and Section 5.3.

**Figure 7-8. Printer options dialog**



On Windows there is a "Details..." button next to the Printer check box that will allow you to select if the print job should be sent to the default printer or if the printer selection dialog should be used on every print.

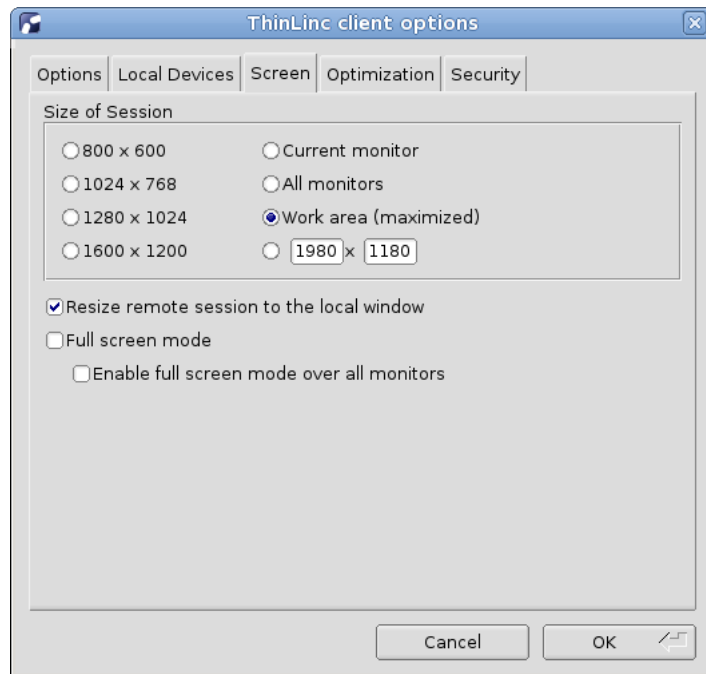
### Export - Smart Card Readers

This check box makes all local smart card readers and smart cards available to applications on the ThinLinc server. It is not necessary to check this box to authenticate using smart cards, but it is needed if you also wish to authenticate using smart cards to a Windows Terminal Services server.

### 7.4.3. Screen tab

The "Screen" tab contains options regarding the session screen. This includes initial screen size, resize behaviour and full screen mode.

**Figure 7-9. Client settings Screen tab**



### Description of screen tab settings

Here follows detailed description of the settings available in the screen tab.

#### Size of session

In this box of radio buttons you can select the screen size you want for your ThinLinc session. The first five options are static with four very common screen sizes (800\*600, 1024\*768, 1280\*1024 and 1600\*1200).

The option *Current monitor* makes the ThinLinc session just as large as the monitor that the main client window is currently on. This can be used to run ThinLinc in full screen mode on one monitor, whilst retaining access to the local desktop on the other monitors.

The option *All monitors* makes the session large enough to cover all available monitors. This is a good choice when using *full screen mode*.

The options *All monitors* and *Current monitor* are identical if there is only a single monitor connected.

The option *Work area (maximized)* makes the ThinLinc session size suitable for a maximized window.

The final possible size option is to manually enter the wanted width and height. The two text boxes close to the last radio button is supposed to contain the width and the height of the wanted session as numbers. These numbers must be larger than 128 and not larger than 16384.

#### **Resize remote session to the local window**

This option makes the remote session follow the size of the local ThinLinc Client window. If the local window is resized, the remote session will be adjusted to match. If this option is disabled, or if the server is too old, padding or scroll bars will be added as needed when the remote session does not match the size of the local window.

#### **Full screen mode**

This option enables *full screen mode* during sessions. That means that the ThinLinc session will cover all of the screen area. If the session is smaller than your screen resolution, there will be black borders around your session which will be centered on the screen. If you run in full screen mode and want to reach the native session that is hidden by the ThinLinc session you can switch out from full screen mode. To do this you press the key assigned to bring up the session pop-up menu. Normally this menu is bound to the F8 key, but can be manually changed. See the *Popup menu key* setting on the *Options tab* above for more information on this. In the session menu you should select *Full screen* to toggle full screen mode.

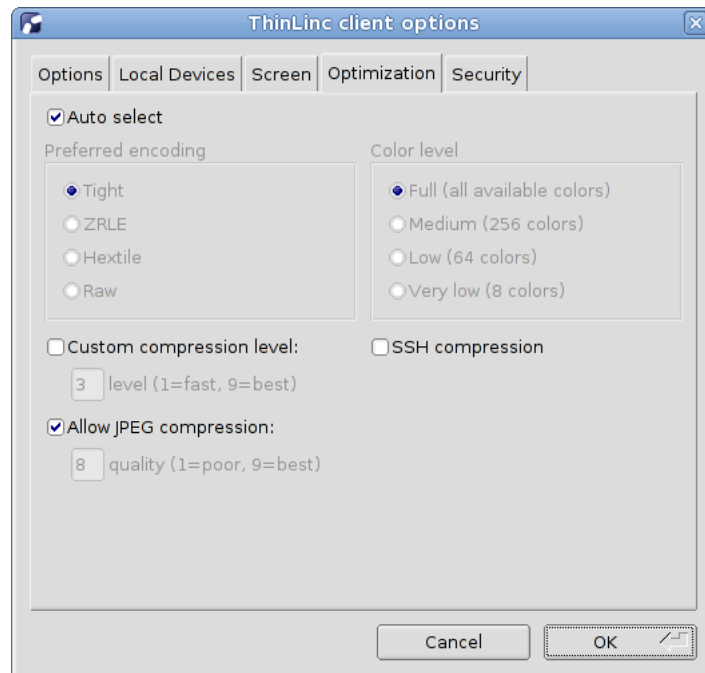
#### **Enable full screen mode over all monitors**

This option controls if *full screen mode* should use just the current monitor, or all monitors connected to the local client.

### **7.4.4. Optimization tab**

The "Optimization" tab contains various settings that affects the protocols used to transfer the graphic information. This includes algorithm used for the graphic encoding. The best choices may differ from case to case. Factors that affects the algorithm choices can for example be network bandwidth, network latency, and client computer performance.

The default setting is to use the *Auto select* mode, to automatically select the best suited algorithms.

**Figure 7-10. Client settings Optimization tab**

## Description of optimization tab settings

Here follows detailed description of the settings available in the optimization tab.

### Auto select

This option makes the ThinLinc system try to automatically select the best suited encoding algorithm. The network performance is measured during the life of the connection and the encoding options are adjusted based on the results. This means that the encoding options can be changed automatically during the connection, if the network performance changes. Activating this option will "gray out" the *Preferred encoding* and *Color level* options, to show that they aren't manually controlled.

### Preferred encoding

This block of settings affects the VNC protocol encoding. There are several different ways to compress and encode the graphic data that is sent from the server to your client. In this box you select one of four possible encodings. The methods differ much: Some try to use smart algorithms to select and compress the areas to send, which means slightly higher CPU usage, but most likely less bandwidth usage and faster sessions where the bandwidth is limited. Other methods use less CPU capacity but more network bandwidth. The best choice can vary much depending on place and

situation. A safe choice is to let the system automatically select the best encoding by checking the *Auto select* checkbox above.

#### **Encoding: Tight**

This choice selects the Tight encoding method. With this encoding the *zlib* compression library is used to compress the pixel data. It pre-processes the data to maximize compression ratios, and to minimize CPU usage on compression. Also, JPEG compression may be used to encode color-rich screen areas. The *zlib* compression level and the JPEG compression ratio can be manually changed. See *Custom compression level* and *Allow JPEG compression* below. *Tight encoding is usually the best choice for low-bandwidth network environments (e.g. slow modem connections).*

#### **Encoding: ZRLE**

This choice selects the ZRLE encoding method.

#### **Encoding: Hextile**

This choice selects the Hextile encoding method. With Hextile the screen is divided into rectangles, split up in to tiles of 16x16 pixels and sent in a predetermined order. *Hextile encoding is often the best choice for using in high-speed network environments (e.g. Ethernet local-area networks).*

#### **Encoding: Raw**

This choice selects the Raw encoding method. This is the simplest of the encoding methods. It simply sends all the graphic data of the screen, raw and uncompressed. *Since this method use the least processing power among the possible methods this is normally the best choice if the server and client runs on the same machine.*

#### **Custom compression level**

By selecting this option you choose to override the standard compression level used when compressing data with the Tight encoding. You can manually select the wanted compression level by entering a number between *1 and 9*. Level 1 uses a minimum of CPU performance and achieves weak compression ratios, while level 9 offers best compression but is slow in terms of CPU consumption on the server side. Use high levels with very slow network connections, and low levels when working over high-speed network connections. *This applies to the Tight encoding only!*

#### **Allow JPEG compression**

By selecting this option you choose to override the standard JPEG compression quality of color-rich parts of the screen. JPEG is a "lossy" compression method for images that helps the Tight encoding to significantly reduce the size of the image data. The drawback is that the resulting image, depending of selected compression ratio, can be blurred and grainy. You can manually select the wanted image quality by entering a number between *0 and 9*. Quality level 0 gives bad image quality but very impressive compression ratios, while level 9 offers very good image quality at lower compression ratios. Note that the Tight encoder uses JPEG to encode only those screen areas that look suitable for lossy compression, so quality level 0 does not always mean unacceptable image quality.

#### **Color level**

This block of choices selects the number of colors to be used for the graphic data sent from the server to the client. The setting has four levels, *Full*, *Medium*, *Low* and *Very low*. The default and



normal is to use the *Full* setting. Selecting a lower number of colors will highly affect the resulting image to the worse, but may also speed up the transfer significantly when using slow network connections.

In this context, *Full* means the number of colors supported by the clients graphics hardware.

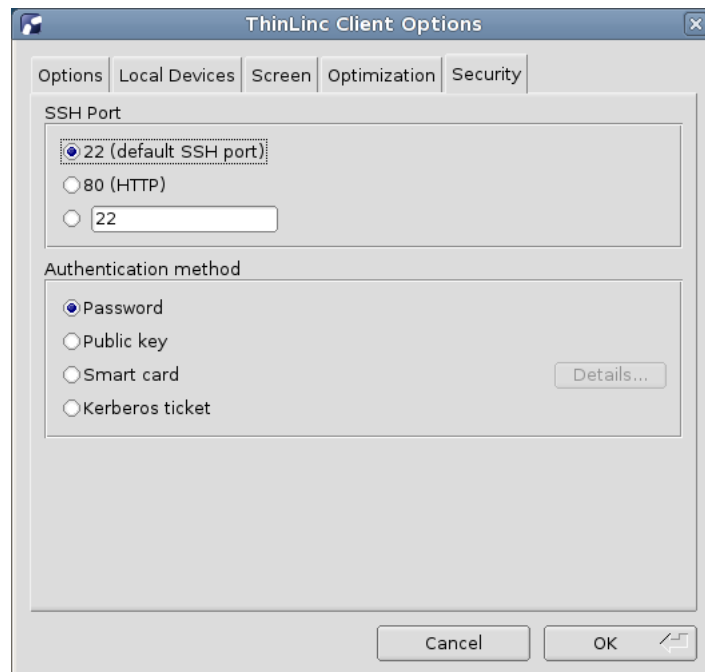
### SSH compression

This choice selects whether or not to use SSH compression for all the data sent between ThinLinc server and client. This is normally not used since an extra compression step, above a compressing graphic encoding normally doesn't help making it smaller, only use more CPU performance. There can still be occasions where this is worth trying though. It is possible that this can help speeding up printing or other exports over slow connections.

## 7.4.5. Security tab

The "Security" tab controls how the client authenticates against the ThinLinc server. The main interface of the client will be different depending on the choices made here.

**Figure 7-11. Client settings Security tab**



## Description of security tab settings

Here follows detailed description of the settings available in the security tab.

### SSH Port

This option selects the TCP/IP port to use when the client tries to establish an SSH connection with the ThinLinc server. The normal SSH port is 22, which also is the default selection for this option. There can be reasons to use another port on some occasions. If you for example need to use ThinLinc over the Internet, from a location where port 22 is blocked by a firewall. Then you can select a port that is let open. Port 80 which is used for HTTP, the protocol used for transport when surfing the WWW is one port that often is open. To be able to use a port other than 22 you need to make sure that the SSH daemon (sshd), which runs on the ThinLinc server, listens to the port you want to use. The SSH daemon can be told to listen to any wanted ports. In the client interface you can select between the default port 22, port 80 and an arbitrary port number which you can enter by yourself.

**Note about SSH host key updates:** If the SSH host key on the server changes, e.g. due to an upgrade of the OS or SSH server software, the client will note this fact. It will then, at the next login, open a dialog and let the user confirm that the new host key is valid. If the user clicks OK , then the host key on the client for this particular server is updated on disk.

The administrator can disallow this by manually setting the parameter `ALLOW_HOSTKEY_UPDATE` to 0. See Section 7.7 for more information.

### Password

This option makes the client try to authenticate using a regular password.

### Public key

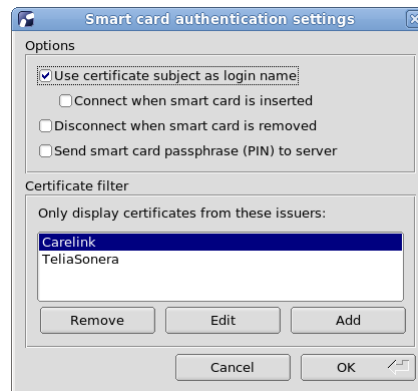
This option makes the client try to authenticate using public key encryption. The user will be asked to provide a private encryption key instead of a text password.

### Smart card

This option makes the client try to authenticate using public key encryption, but with the private key securely stored on a smart card. The user will be asked to select a certificate on the smart card and to provide the passphrase for it.

**Note:** Smart card authentication requires that the smart card is readable by your PKCS#11 library. The library included by default supports PKCS#15 compliant smart cards and relies on the PC/SC interface. This is always present on Windows systems and is usually installed by default on Linux systems. On Solaris this is a third-party add-on.

The "Details..." button lets you change the options for smart card usage and managing the certificate filters which are used to match accepted certificates for authentication.

**Figure 7-12. Smart card authentication settings****Use certificate subject as login name**

Enable this options if you want to enable automatic login, this will also hide the input box for login name from user.

**Connect when smart card is inserted**

This options will try to automatic connect and is dependent on certificate filters, automatic connect will only occur if only one certificate is available after the filtration.

Read more about automatic connection below where certificate filters is discussed.

See Section 9.5.6 for information on how to configure the server for automatic smart card connection.

**Disconnect when smart card is removed**

Enabling this options makes the client automatically disconnect when the smart card used to authenticate is removed.

**Send smart card passphrase (PIN) to server**

This option makes the client transmit the smart card passphrase, as entered by the user, over to the ThinLinc server. It is required to enable smart card single sign-on.

**Warning**

Enabling this option reduces the security of the smart card as the passphrase would otherwise never leave the client system. The option should be left disabled if smart card single sign-on is not used.

## Smart card - certificate filter

A certificate filter is used to present only allowed certificates for authentication, certificates that does not match any filter will be hidden from the user.

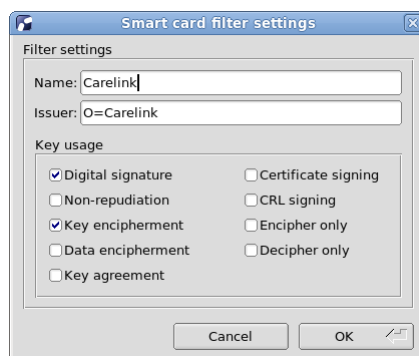
When no certificate filters are configured, all available certificates on the smart card will be available for authentication and therefore the autoconnect feature will not work.

If the resulting filtered list of certificate evaluates only one certificate for authentication and the autoconnect feature is enabled, it will be used for authentication.

When the login dialog is displayed and the key shortcut **control-shift-F8** is pressed, the certificate filtering functionality is bypassed and gives you access to all certificates available on the smart card for authentication.

To add a new filter just press the add button as shown in dialog Figure 7-12 or select an available filter item in the list and press edit to change the settings for specific filter. Either way, the certificate filter settings dialog Figure 7-13 will be shown where you can modify the settings of the specific filter.

**Figure 7-13. Certificate filter settings**



### Name

Enter name of the filter that will be seen in the list of filters.

### Issuer

The certificate issuer field consists of a comma separated list of attribute-value pairs that all must be present in the certificate issuer field. Commonly the "common name" of the issuer is used, e.g. "cn=My CA". It is also possible to allow any issuer that are part of the same organisation, e.g. "o=My Company Ltd.". Any registered object identifier descriptor can be used as an attribute name (see IANA (<http://www.iana.org/assignments/ldap-parameters>) for a full list).

**Key usage**

The certificate must have all the key usage bits selected in this window. Having more bits than those selected does not exclude a certificate.

**Kerberos ticket**

This option makes the client try to authenticate using an existing kerberos ticket.

**Note:** This requires that a valid kerberos ticket is available on the client, and that the SSH daemon on the ThinLinc server is configured to accept this ticket during authentication. For information about how to configure kerberos authentication on your particular platform(s), please see the relevant vendor documentation.

## 7.5. The XDM mode (UNIX only)

When installing dedicated clients, for example old PCs or thin terminal boxes, it's common to install the client to run in XDM mode. XDM is an acronym for X Display Manager and is the name of a small graphical program used for graphical logins in many UNIX systems. By using the ThinLinc client in XDM mode you can make sure that the client appears automatically when the client hardware is started and that it reappears directly after a user logs out.

To run the client in XDM mode you need to start it with the `-x` option. When running in XDM mode the following changes will be made to the client interface.

- The Exit button is removed.
- Control panel is enabled. This makes it possible to control the dedicated client, for example keyboard and mouse.
- Mouse acceleration and keyboard repetition settings are loaded from the client configuration file and applied at startup.
- Always use fullscreen
- More limited F8 menu

### 7.5.1. The XDM mode Control Panel

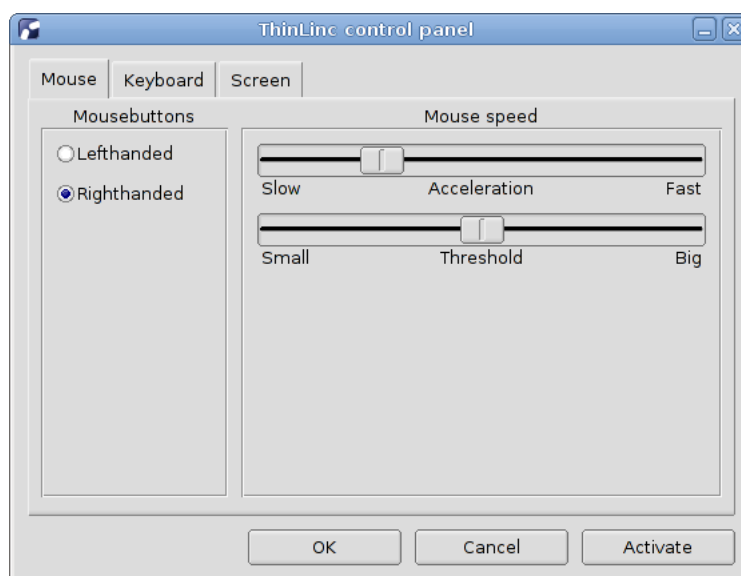
When running in XDM mode a *Control Panel* button will be shown in the ThinLinc login window. When this button is clicked the client will show a control panel where the user can control the *keyboard, mouse and screen*. The reason for this control panel is that the user otherwise has few possibilities to change those settings, since the client runs without any graphical window manager with settings for these options.

The control panel consists of a window with three pages ordered by tabs. One page is used for mouse settings, one for keyboard and one for screen.

### 7.5.1.1. Mouse tab

This page contains mouse settings. The user can here control whether the mouse buttons should be setup for left or right hand use. When changed the left and right mouse button change place. The mouse speed has two controls, the mouse *acceleration* and the *acceleration threshold*. The acceleration setting controls how much the mouse movements will accelerate when the mouse is moved faster. The threshold controls how fast the mouse needs to be moved before the movement will be accelerated. Acceleration is the phenomena where the mouse pointer is moved more if the mouse is moved fast than if it's moved the same distance slowly.

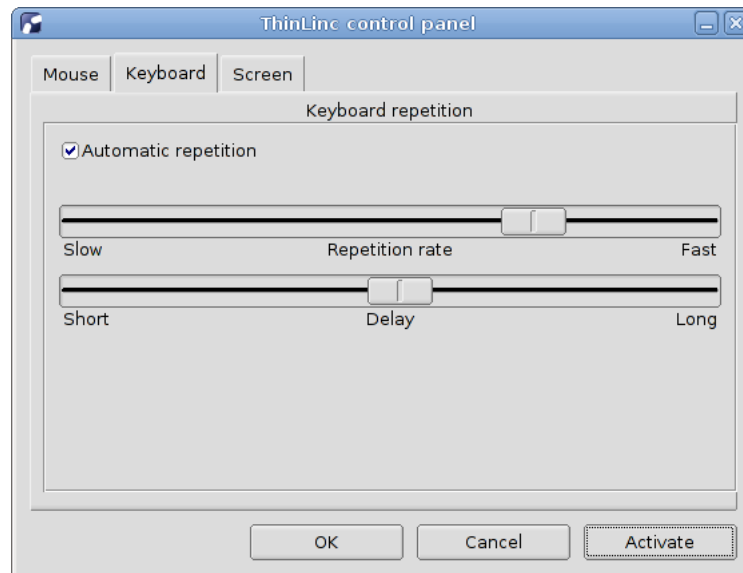
Figure 7-14. The control panel mouse tab



### 7.5.1.2. Keyboard tab

The keyboard tab contains settings for keyboard repetition. The check box *Automatic repetition* controls whether a key that is held down should be repeated automatically or not. The other two controls sets the *repetition rate* and the *repetition delay*. The repetition rate is the speed of the repetition, how many chars from a held down key that will be written each time interval. The repetition delay controls how long the key needs to be held down before it becomes repeated. A too short delay can be bad since it then becomes possible to write double characters by mistake.

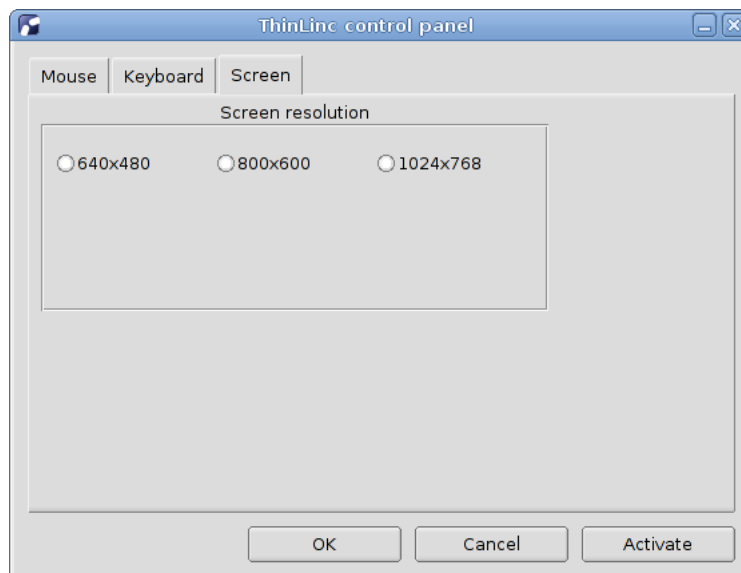
**Figure 7-15. The control panel keyboard tab**



### 7.5.1.3. Screen tab

The screen tab makes it possible to select between different available screen resolutions. The list of possible resolutions is read from the settings for the current terminal.

**Figure 7-16. The control panel screen tab**



## 7.6. Logfile placement

The ThinLinc client logs its activities to the human readable file, `tlclient.log`. The locations of this file differs between UNIX and Windows systems and are explained below.

The log file is truncated every time the ThinLinc client is started. This means that the log file always contains information about the latest session only.

### 7.6.1. UNIX log file

On UNIX systems the logfile is located in the home directory for the user that runs the ThinLinc client. The path is `~/.thinlinc/tlclient.log`.

### 7.6.2. Windows log file

On Windows systems the logfile is located in the directory referenced from the `%TMP%` variable. The value of this variable can be achieved by running any of the following commands in a Windows command window.

```
C:\> echo %TMP%
```

or



```
C:\> set
```

Observe that some directories in the Windows %TMP% path may be flagged as hidden by the Windows system. This means that you need to change directory options to display hidden files and directories to navigate to the log file.

## 7.7. Client configuration storage

### 7.7.1. Overview and Parameters

The ThinLinc client does not currently use Hiveconf for its configuration, although this may change in future ThinLinc releases. Instead, the Linux client uses a similar plain text format with key/value pairs and the Windows client stores the values in the Windows registry.

**Note:** The configuration parameters should seldom be edited by hand. For a system wide configuration, create a parameter set using the client and copy it to the system wide file.

### Configuration Parameters Used by the ThinLinc Client

Both the Windows and the Linux version of the ThinLinc client use the same names for their configuration parameters, although the storage technique used is different (text files vs. registry keys). In this section we will list the parameters and explain their possible values.

#### ALLOW\_HOSTKEY\_UPDATE

Set to 1 if SSH host key updates should be allowed. This parameter cannot be changed from the GUI. The result of setting ALLOW\_HOSTKEY\_UPDATE to 0 is that the client cannot connect to the server if the host key is wrong. This enhances security if there is a risk for a man in the middle attack.

#### AUTHENTICATION\_METHOD

This parameter can be set to "password", "publickey", "scpupublickey" or "kerberos" to select the authentication mode used by the client.

#### AUTOLOGIN

If this parameter is set to 1, the client will automatically login at start, using the server name, user name and password specified in the configuration storage.

#### CERTIFICATE

Specifies the smart card certificate to use when authenticating.

#### CERTIFICATE\_NAMING

Controls how the client presents a certificate to the user. The parameter consists of a comma separated list of naming tokens that represent bits of information from each card or certificate. Possible tokens:

`card_label`

The label specified on the smart card.

`pin_label`

The label associated with the PIN protecting this certificate.

`subject_*`

A field from the subject in the certificate. Can for example be the common name by specifying *subject\_cn* or *subject\_commonName*. Any registered object identifier descriptor can be used (see IANA (<http://www.iana.org/assignments/ldap-parameters>) for a full list).

`issuer_*`

A field from the issuer in the certificate, in the same manner as for *subject\_\**.

The client will use as many of the tokens as necessary to give each certificate a unique name. That means that certificates on two different cards can be presented with a different number of tokens depending on how much the information between the certificates overlap. An index number will be added to the name if the names are still not unique when all tokens are used.

`CUSTOM_COMPRESSION`

Set to 1 if a custom compression method is selected.

`CUSTOM_COMPRESSION_LEVEL`

The selected compression level. An integer between 1 and 9.

`DISPLAY_MODE`

The display mode. Can be set to values "SIMPLE" and "ADVANCED", or be left empty. In the latter case, the default behaviour is to use simple mode if a server name is given as a parameter and advanced mode otherwise.

`REMOTE_RESIZE`

Set to 1 if the client should resize the remote session when the local window changes.

`FULL_SCREEN_MODE`

Set to 1 if the client should run in fullscreen mode.

`FULL_SCREEN_ALL_MONITORS`

Set to 1 if the client should use all monitors in full screen mode, instead of just the current monitor.

`HOST_ALIASES`

This parameter specifies a list of hostname and port translations. This translation list is consulted whenever the client is about to initiate a network connection. This includes the SSH connection to the ThinLinc agent machine. The syntax for this parameter is:

```
[fromhost][:fromport]=[tohost][:toport] ...
```

If `fromhost` is omitted, the translation will apply to all hosts. The same principle is used for ports. If `tohost` or `toport` is omitted, the original host or port will be used. Multiple translations are separated with whitespace. The translation stops as soon as one match is found.

#### JPEG\_COMPRESSION

Set to 1 if JPEG compression is wanted.

#### JPEG\_COMPRESSION\_LEVEL

The wanted compression level.

#### KILL\_EXISTING\_SESSIONS

Set to 1 if existing sessions should be ended.

**Note:** It makes little sense to change this value. The client never saves this setting.

#### LOWERCASE\_LOGIN\_NAME

Set to 1 if the client should convert the entered username to lowercase before logging into the server. This affects both the login user name and the name of the user to shadow (if applicable).

#### NEW\_PASSWORD\_REGEX

This parameter specifies a regular expression. If an interactive SSH prompt matches this expression, the response is taken as a new password. The new password will be used for the SSH connection to the agent machine. It will also be sent to the server to enable Single Sign-On.

#### OPTIONS\_POPUP\_KEY

Key code for key to activate option pop-up menu.

#### PASSWORD

This parameter allows you to specify a password in the configuration file. It must be specified using a hexadecimal ASCII notation, which means that every character is specified by its hexadecimal value.

### Warning

The password value is not encrypted. It should be treated as a clear text password. Avoid storing configuration files with a `PASSWORD` parameter on disk or transmit such files over networks without encryption.

#### PKCS11\_MODULE

Specifies the PKCS#11 module that will be used to communicate with the smart card. The path can be relative the base prefix of the ThinLinc client or an absolute path.

PRIVATE\_KEY

This parameter specifies the path to the private key to be used to authenticate the user.

RECONNECT\_POLICY

This parameter can be set to "single-disconnected" or "ask" to control the client's reconnect policy. See Section 7.4.1 for their meaning.

REMOVE\_CONFIGURATION

If 1, the user configuration file (or the file specified by -C) will be removed after the client has started. Settings changed in the GUI will not be stored to disk. If the client fails to remove the file, it will try to truncate it instead.

SCREEN\_SIZE\_SELECTION

The default size of the ThinLinc session. Possible values:

- 0 for 640x480
- 1 for 800x600
- 2 for 1024x768
- 3 for 1280x1024
- 4 for 1600x1200
- 5 for Current monitor
- 6 for Work area (maximized)
- 7 for Custom screen size, set using the SCREEN\_X\_SIZE and SCREEN\_Y\_SIZE parameters.
- 8 for All available monitors

SCREEN\_X\_SIZE

Custom width of session, if SCREEN\_SIZE\_SELECTION is set to 7.

SCREEN\_Y\_SIZE

Custom height of session, if SCREEN\_SIZE\_SELECTION is set to 7.

SEND\_SYSKEYS

Set to 1 if the client should send system keys (like Alt+Tab) to the remote system when in full screen mode.

SMARTCARD\_FILTER\_n

This is a item list of certificate filters replace *n* with a sequence number that defined the order of the filter in the list.

The filter string consists of three fields where each field is sperated using a | (pipe), the defined three fields are: name, attributes and key usage which are documented below. Here follows an example of a filter string showing its format:

```
SMARTCARD_FILTER_1=Telia|o=TeliaSonera|5
```

**name**

The name of the filter which will be displayed in the list of filters defined in the user interface.

**attributes**

This field holds a comma separated list of certificate attributes that is used when matching against available certificates, for example *O=TeliaSonera*.

**key usage**

Key usage is a bitmask value used to match against a certificate's key usage flags. It indicates the intended usage of the certificate, such as identification, signing etc.

Use this to match certificates that is intended to be used for logon. For example, identification certificates will be matched using a value of 5, *digital signature + key encipherment* = 5. The values are described in the following table:

1	digital signature
2	non-repudiation
4	key encipherment
8	data encipherment
16	key agreement
32	certificate signing
64	CRL signing
128	encipher only
256	decipher only

**SMARTCARD\_SUBJECT\_AS\_NAME**

Set to 1 if the certificate subject should be used as logon name, this will hide the name field from login window.

**SMARTCARD\_AUTOCONNECT**

Set to 1 if the client should automatically attempt a connection when a smart card with a suitable certificate is found, this will only work if SMARTCARD\_SUBJECT\_AS\_NAME also is set to 1.

**SMARTCARD\_DISCONNECT**

Set to 1 if the client should disconnect automatically when the smart card used for authentication is removed.

**SMARTCARD\_PASSPHRASE\_SSO**

Set to 1 if the client should transmit the smart card passphrase to the ThinLinc server to enable smart card single sign-on. See Section 7.4.5 for security implications.

**SOUND\_ENABLED**

Set to 1 if sound redirection should be enabled.

SOUND\_SYSTEM

Which local sound system to use. Only used on platforms that have multiple sound systems to choose from. Possible values:

AUTO

Automatically choose the most appropriate sound system of those available.

PULSE

Use the local PulseAudio server as determined by X11 properties or environment variables.

ALSA

Use the default ALSA device.

OSS

Use the default OSS device.

SSH\_PORT\_SELECTION

Port selection for ThinLinc connection. Possible values:

- 0 for port 22 (standard ssh port).
- 1 for port 80.
- 2 for custom port set in the SSH\_ARBITRARY parameter.

SSH\_ARBITRARY

Custom port number for ThinLinc connection.

TUNNEL\_BIND\_BASE

Offset for user's forwarded ports, on server side.

**Note:** This parameter should normally not be changed

START\_PROGRAM\_ENABLED

Specifies if the client should request that the server starts the session with the command supplied by the client.

START\_PROGRAM\_COMMAND

Specifies the command to use when starting the session, if START\_PROGRAM\_ENABLED is active.

TUNNEL\_AUDIO\_SLOT

Port selection for audio tunnel, on server side.

**Note:** This parameter should normally not be changed.

VNC\_AUTOSELECT

Set to 1 to dynamically autoselect the compression algorithm during the session.

VNC\_ENCODING\_SELECTION

The encoding to use for VNC. Possible values:

- 0 for Raw
- 5 for Hextile
- 7 for Tight
- 16 for ZRLE

VNC\_COLOR\_LEVEL

The color level used for the session.

USE\_SSH\_COMPRESSION

Set to 1 to use the compression built into SSH.

UPDATE\_ENABLED

Set to 1 to enable periodic checks for new versions.

UPDATE\_INTERVAL

This parameter specifies the time interval, in seconds, between client update checks.

UPDATE\_LASTCHECK

This parameter specifies the time that the last update check was performed.

UPDATE\_MANDATORY

If set to 1, updating to new client versions is mandatory.

UPDATE\_URL

The HTTP URL to client update configuration file.

VNC\_PORT\_BASE

Offset for Xvnc VNC ports on server side.

**Note:** This parameter should normally not be changed

VSM\_SERVER\_PORT

The VSM port number on the server.

**Note:** This parameter should normally not be changed

YESNO\_PROMPT\_REGEX

This parameter specifies a regular expression. If an interactive SSH prompt matches this expression, a graphical yes/no dialog will be presented, instead of a dialog for text input. Additionally, if the prompt is known to the client, an alternate text will be used. The dialog buttons Yes and No will send "yes" and "no" to the server, respectively.

## 7.7.2. Configuration Parameter Storage

Configuration parameters are typically stored in text based configuration files. The format is simple: Each parameter is written on one line, followed by an equal sign (=) and the value of the parameter, as in the following example:

```
SOUND_ENABLED = 0
SERVER_NAME = demo.thinlinc.com
```

By using the -C option, additional configuration files can be specified. Any name is accepted, but the file extension `.tlclient` is recommended. The Windows and Linux packages configure the system to automatically recognize such files as configuration files for the ThinLinc Client. Additionally, the Internet Media Type "application-vnd.cendio.thinlinc.clientconf" is linked to such configuration files.

### 7.7.2.1. Linux Client Configuration Files

The Linux client first reads the file `/opt/thinlinc/etc/tlclient.conf`, if it exists. It then reads the file `.thinlinc/tlclient.conf` in the user's home directory, and the values there override the values from `/opt/thinlinc/etc/tlclient.conf`. This way, a system administrator can set global defaults for client operations, while each user can still customize the client to wanted behavior.

### 7.7.2.2. Windows Client Configuration

On Windows, the ThinLinc client reads its configuration from the registry. All ThinLinc client data is stored under `Software\Cendio\ThinLinc\tlclient` in the HKLM and HKCU hives. The parameter names are the same as for the Linux client.

The behaviour of global and user-specific settings are identical to that of the Linux client, where settings in HKLM correspond to `/opt/thinlinc/etc/tlclient.conf` and those in HKCU correspond to `.thinlinc/tlclient.conf`.



### 7.7.3. Adding Custom Branding to the ThinLinc Client Login Window

It is possible to add a custom logo to the main ThinLinc client window, making it easily distinguishable from a generic client. The custom logo will be placed to the right of the input fields.

Adding the logo is easy. The new logo must be a PNG file with maximum width and height of 50 pixels. On Windows, just add the file `branding.png` in the same directory as the executable with the custom logo. On Linux, the file name is `/opt/thinlinc/lib/tlclient/branding.png`.

## 7.8. Client Customizer

### 7.8.1. Introduction

This software lets you create customized ThinLinc client installation programs. This means that when users install the customized version, they will automatically get the default settings you have configured.

One advantage with this is that you can provide, for example, a default server name. A custom client can also be used to enhance security: You can distribute SSH host keys with the client itself, so that users don't need to be concerned with SSH host key fingerprint verification.

**Note:** The Client Customizer only works for the Windows client.

### 7.8.2. Installation

Before you can start, you have to install the build environment. This is done by running the command **tl-4.3.0-client-customizer.exe** located in the Client Bundle. This will also create a shortcut to the build directory in the Start menu.

### 7.8.3. Building a Customized Client

To create a customized client, do the following:

1. Edit `settings.reg`. This file contains all the parameter names and default values that are used in **tlclient**. To customize the client, edit any of these values, and they will be installed in the registry when you install the customized client itself. You can also add your servers SSH host keys (see below).
2. Run **build.bat** in the same directory. The file `setup.exe` will now be created. This is the installation program for the customized client.

### 7.8.4. Adding SSH Host Keys to `settings.reg`

To add your servers SSH host key to `settings.reg`, do the following:

1. Use **tlclient** to connect to your server, if you haven't already done so. Confirm the servers host key, if necessary.
2. Run the registry editor, and select  
`HKEY_CURRENT_USER\Software\Cendio\ThinLinc\tlclient\KnownHosts`
3. Export this key to an external text file.
4. Open the text file from the previous step in an editor.
5. Copy the line corresponding to your ThinLinc server. Paste this line into `settings.reg`, section  
`HKEY_LOCAL_MACHINE\Software\Cendio\ThinLinc\tlclient\KnownHosts`
6. Save `settings.reg`, and proceed to create the customized client as described above.

## 7.9. Advanced Topics

### 7.9.1. Hardware Address Reporting

When the client connects to server, it reports it's hardware address. On Linux, the active interface with the smallest MAC address is used. On Windows, the address of the first interface (as listed in the Control Panel) is used. On Solaris, the reported address is CFFF followed by the hostid.

### 7.9.2. Client Update Notifications

The client includes a feature which can periodically check for new versions. This functionality is affected by the configuration parameters `UPDATE_ENABLED`, `UPDATE_INTERVAL`, `UPDATE_LASTCHECK`, `UPDATE_MANDATORY`, and, `UPDATE_URL`. These are described in Section 7.7. During an update check, the client retrieves the file specified by `UPDATE_URL`. An example follows:

```
WINDOWSINSTALLER = https://www.cendio.com/downloads/clients/tl-latest-client-windows.exe
LINUXINSTALLER = https://www.cendio.com/downloads/clients/thinlinc-client-latest.i686.rpm
DEFAULTINSTALLER = https://www.cendio.com/downloads/clients
OKVERSIONS = 3.2.0 3.3.0
```

The `OKVERSIONS` parameter specifies a list of valid client versions. If the running client version is different, the client will notify the user. The `WINDOWSINSTALLER`, `LINUXINSTALLER`, and `DEFAULTINSTALLER` parameters specifies the updated client packages for Windows, Linux, and other platforms, respectively.

## Chapter 8. Client Platforms

There are several ways to run the ThinLinc client, and also some ways to access ThinLinc servers without running the client.

In this chapter we will document how to install, configure and run the ThinLinc client on all different platforms including dedicated thin terminals.

### 8.1. Windows

#### 8.1.1. Requirements

The supported Windows versions are XP, XP Embedded, 2003, 2003 R2, Vista, 2008, 7, 2008 R2, 8, 2012, 8.1, 2012 R2. Windows CE is currently not supported.

#### 8.1.2. Installing the Windows Client

To install the client on a Windows machine, unpack the Client Bundle and enter the `client-windows` directory. Then click on the file `tl-4.3.0-client-windows.exe` and follow the instructions.

The `tl-4.3.0-client-windows` directory contains an unpacked version of the ThinLinc Windows client. It makes it possible to run the client directly from the bundle, without installing first.

For more information about how to configure the client, read Section 7.7.

#### 8.1.3. Running the Windows Client

During installation the ThinLinc client will be added to the Start menu. To start the client you select it from the Start menu.

### 8.2. Mac OS X

#### 8.2.1. Requirements

- Mac OS X 10.4 or newer running on Intel hardware

**Note:** Mac OS X version 10.9 installs with a default setting that breaks the multi monitor functionality of the ThinLinc client. A workaround to this problem is to disable setting "Displays have separate Spaces" in settings for "Mission Control" found in "System Preferences" .

### 8.2.2. Installing the Mac OS X Client

The client for Mac OS X can be found in the directory `client-osx` in the Client Bundle. To install the client, follow these steps:

1. Double-click on the file `tl-4.3.0_4538-client-osx.iso`.
2. Drag the "ThinLinc Client" application to an application folder of your choice.
3. Eject "ThinLinc Client".

### 8.2.3. Running the Mac OS X Client

To start the ThinLinc Client, double click on the client application. The client can also be added to and started from the Dock.

### 8.2.4. Command and Alt Keys on Mac OS X

The Alt key (also known as the Option key) behaves very differently on OS X compared to its behaviour on other platforms. It closely resembles the PC AltGr key, found on international keyboards. ThinLinc therefore treats these keys in a special manner on Mac OS X in order to provide a good integration between the client and the remote ThinLinc system.

Whenever either of the Alt keys are pressed, ThinLinc will behave as if AltGr was pressed. The left Command key is used as a replacement for Alt in order to use shortcuts like Alt+F. The right Command key retains its behaviour of acting like the Super/Windows key.

## 8.3. Linux PC

### 8.3.1. Requirements

- Any 32-bit distribution based on GLIBC 2.3.4, or newer. An i686 (or compatible) CPU with MMX and SSE support is required.  
or  
Any 64-bit distribution based on GLIBC 2.5.1, or newer. An x86\_64 (or compatible) CPU is required.  
or  
Any 32-bit distribution based on GLIBC 2.11.3, or newer. An ARMv7 (or compatible) CPU with Thumb-2 and VFP3D16 is required.
- A working Fontconfig configuration, or basic fonts available in `/usr/share/fonts` or `/usr/X11R6/lib/X11/fonts`.
- 32 MiB RAM

### 8.3.2. Installing the Linux Client

The Linux client is distributed in three different kinds of packages. One that can be installed using the RPM package system, one in the DEB package format, and one in compressed tar archive form for any Linux distribution.

If you need more information than mentioned here, read Section 7.7.

In the instructions below, we will assume that you have unpacked your Client Bundle to `~/tl-4.3.0-clients`.

#### 8.3.2.1. RPM-based Installation on RPM-based distributions

The RPM-based client can be found in the directory `client-linux-rpm` in the Client Bundle. It is suitable for systems such as Red Hat, Fedora, SuSE, and Mandrake. Perform the following steps to install it on a 32-bit system:

```
$ cd ~/tl-4.3.0-clients/client-linux-rpm
$ sudo rpm -Uvh thinlinc-client-4.3.0-4538.i686.rpm
```

or the following steps on a 64-bit system:

```
$ cd ~/tl-4.3.0-clients/client-linux-rpm
$ sudo rpm -Uvh thinlinc-client-4.3.0-4538.x86_64.rpm
```

or the following steps on a 32-bit ARM hard-float system:

```
$ cd ~/tl-4.3.0-clients/client-linux-rpm
$ sudo rpm -Uvh thinlinc-client-4.3.0-4538.armv7hl.rpm
```

or the following steps on a 32-bit ARM soft-float system:

```
$ cd ~/tl-4.3.0-clients/client-linux-rpm
$ sudo rpm -Uvh thinlinc-client-4.3.0-4538.armv7l.rpm
```

#### 8.3.2.2. DEB-based Installation on Debian and Ubuntu based distributions

The DEB-based client can be found in the directory `client-linux-deb` in the Client Bundle. It is suitable for systems such as Debian and Ubuntu. Perform the following step to install it on a 32-bit system:

```
$ cd ~/tl-4.3.0-clients/client-linux-deb
$ sudo dpkg -i thinlinc-client_4.3.0-4538_i386.deb
```

or the following steps on a 64-bit system:

```
$ cd ~/tl-4.3.0-clients/client-linux-deb
$ sudo dpkg -i thinlinc-client_4.3.0-4538_amd64.deb
```

or the following steps on a 32-bit ARM hard-float system:

```
$ cd ~/tl-4.3.0-clients/client-linux-deb
$ sudo dpkg -i thinlinc-client_4.3.0-4538_armhf.deb
```

or the following steps on a 32-bit ARM soft-float system (only Ubuntu):

```
$ cd ~/tl-4.3.0-clients/client-linux-deb
$ sudo dpkg -i thinlinc-client_4.3.0-4538_armel.deb
```

### 8.3.2.3. Installation on Other Linux Distributions

A client without any specific package format can be found in the directory `client-linux-dynamic` in the Client Bundle. It is possible to run this client from any directory, even from the unpacked Client Bundle. We generally recommend installing it in `/opt/thinlinc`. Perform the following steps to install the client to `/opt/thinlinc` on a 32-bit system:

```
$ cd ~/tl-4.3.0-clients/client-linux-dynamic
$ sudo mkdir -p /opt/thinlinc
$ sudo cp -a tl-4.3.0-4538-client-linux-dynamic-i686/* /opt/thinlinc/
```

or the following steps on a 64-bit system:

```
$ cd ~/tl-4.3.0-clients/client-linux-dynamic
$ sudo mkdir -p /opt/thinlinc
$ sudo cp -a tl-4.3.0-4538-client-linux-dynamic-x86_64/* /opt/thinlinc/
```

or the following steps on a 32-bit ARM hard-float system:

```
$ cd ~/tl-4.3.0-clients/client-linux-dynamic
$ sudo mkdir -p /opt/thinlinc
$ sudo cp -a tl-4.3.0-4538-client-linux-dynamic-armhf/* /opt/thinlinc/
```

or the following steps on a 32-bit ARM soft-float system:

```
$ cd ~/tl-4.3.0-clients/client-linux-dynamic
$ sudo mkdir -p /opt/thinlinc
$ sudo cp -a tl-4.3.0-4538-client-linux-dynamic-armel/* /opt/thinlinc/
```

The client is also available as tar archives for easy transfer to other systems without having to copy the entire Client Bundle.

### 8.3.3. Running the Linux Client

On Linux and other UNIX systems the client will be installed as `/opt/thinlinc/bin/tlclient`. The client package contains settings that adds `/opt/thinlinc/bin` to the path of the users.

To run the client, click on the "ThinLinc Client" icon in your desktop environment. Typically, the icon is found in the Internet category. You can also run the client by executing `/opt/thinlinc/bin/tlclient`.

## 8.4. Solaris

The Solaris client is very similar to the Linux client.

### 8.4.1. Requirements

- Sun Solaris® 10 on SPARC. When using redirected local drives, the package SUNWscpu needs to be installed.

### 8.4.2. Installing the Solaris Client

The Solaris client can be found in `client-solaris/` in the Client Bundle and is shipped as a Solaris package. Install by using the **pkgadd** command.

```
# pkgadd -d CENDthinlincclient-4.3.0-4538-sparc
```

If you would like to install the client as non-root, you can extract the files with the **pkgtrans** command:

```
$ pkgtrans CENDthinlincclient-4.3.0-4538-sparc
```

The client can be installed and run from any location on the filesystem.

### 8.4.3. Running the Solaris Client

For instructions on how to run the Solaris client, see the corresponding instructions for the Linux client in Section 8.3.3.

## 8.5. Thin Terminals

ThinLinc has support for several thin terminals, i.e. hardware built with the task of providing a thin client as primary design goal.

### 8.5.1. Neoware Terminals

The recommended way of running the ThinLinc client on these terminals is to migrate to the "eLux" platform. See Section 8.5.2 for more information. Neoware was acquired by HP in 2007, and no new terminals are produced. Old terminals should work but with limited performance.

### 8.5.2. eLux-based Thin Terminals (Fujitsu Futro et. al.)

ThinLinc has support for running the client on eLux-based thin terminals. This includes the Fujitsu Futro as well as NEXTerminal terminals such as the NEXceed. eLux is a modern embedded operating system which works well.

ThinLinc supports models that uses the eLux RL, RP, or RT operating system. Microphone devices are known to work at least on the Futro models. However, make sure to disable "Sound in XDMCP sessions".

### 8.5.2.1. Installing/Upgrading the ThinLinc Client on eLux

Below we will describe how to install and configure the ThinLinc client on terminals running eLux. For details on how to use the administrative tools (Scout and ELIAS) as well as on how to manually configure terminals without use of Scout, please refer to the eLux documentation available at <http://www.mylux.com>.

1. Create a new firmware image including the ThinLinc client. This is done by copying the files `thinlinc-client-*` from the appropriate container directory in the Client Bundle to your ELIAS/Scout container directory.

**Note:** Some eLux distributions already include the ThinLinc client, rendering this step unnecessary.

2. Add the ThinLinc client to your firmware image using ELIAS.
3. Install the image on your clients using either Scout (for centrally-managed clients), or via http (for single clients).
4. Configure your clients by adding an application of type *Custom* under the *Local* tab under *Configuration* on the client (or create a new application if using Scout). Set the application name to "ThinLinc" and the command line to `tlclient`. Check the "Application restart" and the "Start automatically after 0 s" checkboxes to make sure the ThinLinc client is restarted after end of session, and that it is started automatically at boot.

**Note:** All commandline parameters accepted by the Native Linux client is also accepted by the client run on eLux (it's the same client). This means that server name, username, lockdown options etc. can be used on the commandline specified when configuring the client. An example commandline instructing the client to start with an empty username and to connect to `<server name>` is provided below.

```
tlclient -u " <server name>
```

This configuration can be done either on the client, under the configuration tab in the starter, or centrally, using Scout, by adding an application in the Applications container in an appropriate organizational unit.

5. Configure the starter not to start automatically by unchecking the checkbox in "Autostart Starter after 0 s" under *Setup Desktop Advanced* . Unchecking the "Enable" button for the Taskbar in the same location may also be a good idea to give users a cleaner environment.

This configuration can be done either on the client, under the Setup tab in the starter, or centrally, using Scout.



6. If custom configuration of the client, for example to support local drives, is needed, transfer a custom client configuration file (tlclient.conf) to /setup/thinlinc/tlclient.conf on the eLux clients using the "Transfer files" functionality available in the properties of devices or organisation units in Scout (under the Files tab). The files are transferred when the client is restarted.

### 8.5.3. VXL

The recommended way of running the ThinLinc client on these terminals is to migrate to the "eLux" platform. See Section 8.5.2 for more information. Note however that only a few VXL terminals are supported by eLux.

### 8.5.4. HP ThinPro Terminals

HP ThinPro terminals are based on Ubuntu, and therefore one can use the DEB package provided in our ThinLinc Client bundle for this terminal. The "armel" package is used on ARM based terminals such as t410. For other terminals, use the "i386" package.

#### 8.5.4.1. Manual installation/upgrade of ThinLinc Client

Below we will describe the process of manually installing the ThinLinc Client on Ubuntu based HP ThinPro Linux terminals.

1. Use the tool "Administrator/User mode switch" to authenticate as administrator.
2. Start an X terminal from the advanced tab in the control panel.
3. Unlock the filesystem:
 

```
# fsunlock
```
4. Copy the ThinLinc Client .deb package from ThinLinc Client bundle onto a USB memory stick, and connect it to the terminal. Go into the directory which represents your connected USB device with command:
 

```
# cd /tmp/tmpfs/media/my_usb_storage
```

 As an alternative, it is also possible to download the client package from a web server using the "wget" command.
5. Install the ThinLinc Client package using Debian package manager command:
 

```
# dpkg -i thinlinc-client*.deb
```
6. Lock down the filesystem before closing the X terminal window:
 

```
# fslock
```
7. Reboot.
8. Add a ThinLinc connection in the connection manager.

The HP "Connection Wizard" does not include an entry for ThinLinc. Press "Skip", then add a ThinLinc Connection in the "Connection Manager".

The default user and administrator share the same home directory, and it is therefore important to NOT start the ThinLinc Client as administrator the first time. This will make the ThinLinc Client configuration only accessible by administrator and not the default user.

On "zero" clients, the default server name is set when the ThinLinc connection type is selected. To change server name, temporarily switch to another connection type, then switch back to ThinLinc. Also, to configure the ThinLinc Client, enter an invalid username/password combination in the HP login dialog. Acknowledge the error. It is then possible to access the full ThinLinc Client interface.

### 8.5.5. IGEL Universal Desktop

A client package for IGEL Universal Desktop terminals is provided. It is included in the directory `client-igel` in the Client Bundle. IGEL Universal Desktop is a modern embedded operating system which works well. Some editions includes a bundled ThinLinc client. We do not recommend this client. Instead, install the client as described below.

#### 8.5.5.1. Installing/Upgrading the ThinLinc Client on IGEL terminals

Below we will describe how to install and configure the ThinLinc client on IGEL terminals, using the "Custom partition". You can use either the Universal Management Suite software running on a separate workstation, or the setup software installed on the terminal. You will need access to a web server which allows you to publish the client files.

1. Edit the configuration of the terminal. Select System, Firmware Customization, Custom Partition.
2. Under the Partition option, make sure that "Enable Partition" is checked. Enter a size, such as "100M". The partition must be at least 20 MiB. The upper limit depends on the hardware used. Make sure that the mount point is /custom.
3. Under the Download option, press the star to create a new data source. Enter the URL to the web server where the ThinLinc client package definition is located. The "armel" package is used on ARM based terminals such as the IZ1 and UD2-LX. For other terminals, use the "i686" package. Example: `http://www.example.com/client-igel/thinlinc-i686.inf`
4. Under Custom Application, press the star to create a new application entry. Use a Session Name such as "ThinLinc".
5. Click on Settings. Enter the Icon name:

```
/custom/thinlinc/icon.png
```

To setup the client to use the terminals normal language, enter this Command:

```
/custom/thinlinc/bin/tlclient
```

To setup the client to use Swedish, use this Command:

```
env LC_ALL=sv_SE.UTF-8 /custom/thinlinc/bin/tlclient
```

6. Press OK to save the configuration.

### 8.5.6. Dell Wyse-Enhanced SuSE Linux Terminals

A client Add-On for Dell Wyse-terminals running "Dell Wyse-Enhanced SUSE Linux" is provided in the Client Bundle, inside the `client-wyse/` directory. For instructions on how to install or upgrade this package, please consult the Wyse Linux documentation (<http://www.dell.com/wyse/>).

The Dell Wyse client bundled is built and packaged for SUSE Linux Enterprise Thin Client 11 Service Pack 2 (sletc11sp2). Due to limitations of the package metadata, a package can only support one service pack version of SLETC11 at the same time.

### 8.5.7. Dell Wyse-Enhanced Ubuntu Linux Terminals

With terminals running Dell Wyse-Enhanced Ubuntu Linux Terminals, one can use the DEB package provided in our ThinLinc Client bundle. The "armel" package is used on ARM based terminals such as T50. For other terminals, use the "i386" package.

#### 8.5.7.1. Manual installation/upgrade of ThinLinc Client

Below we will describe the process of manually installing the ThinLinc Client on Ubuntu based Wyse terminals.

1. Create a new custom connection. Enter command "bash" and select "Run in terminal window".
2. Start a terminal using the new connection.
3. Switch to root with "sudo -i".
4. Copy the ThinLinc Client .deb package from ThinLinc Client bundle onto a USB memory stick, and connect it to the terminal. Go into the directory which represents your connected USB device with command:
 

```
# cd /media/my_usb_storage
```

 As an alternative, it is also possible to download the client package from a web server using the "wget" command.
5. Install the ThinLinc Client package using Debian package manager command:
 

```
# dpkg -i thinlinc-client*.deb
```
6. Close the terminal window.
7. Add a new custom connection. The Command should be set to `/opt/thinlinc/bin/tlclient`.

### 8.5.8. Other Thin Terminals

The ThinLinc client can be made to run on almost any Linux-based Thin Terminal as well as on some Windows-based appliances. Contact Cendio if you need help on a consultancy basis.

## 8.6. Running ThinLinc on a Thinstation terminal

The Thinstation project (<http://thinstation.github.io/thinstation/>) is an opensource thin client Linux distribution that can be booted in many different ways, including entirely over the network on diskless machines and via a LiveCD.

A client package for ThinStation is shipped as part of the ThinLinc client distribution. In this section, we will document how to use and configure this package with Thinstation.

**Note:** To run the ThinLinc client in ThinStation, you need ThinStation version 2.2 or later. However, on ThinStation version 2.4 and later, the ThinLinc client will be downloaded automatically during the build process. Thus the download procedure is only needed for ThinStation 2.2. However, the configuration step is needed for both 2.2 and 2.4.

There are two alternative methods of getting a Thinstation image with the ThinLinc client included. The first one is to use one of the TS-O-Matic servers available. They allow you to build Thinstation images online, and should generally have the ThinLinc client available as an option. The TS-O-Matic servers are available from the Thinstation webpages (<http://thinstation.github.io/thinstation/>). The second is to download the Thinstation distribution, add the ThinLinc client package and then configure and build a Thinstation image manually. This requires access to a Linux box.

Below, we will describe the second method

### 8.6.1. Installing and Building the Package

Begin by downloading and unpacking the Thinstation main distribution available from the Thinstation webpages (<http://thinstation.github.io/thinstation/>).

Enter the Thinstation directory created while unpacking, and unpack the ThinLinc Thinstation client package, found in the `client-thinstation` directory in the Client Bundle, into this directory:

```
$ tar zxvf tl-4.3.0-4538-client-thinstation.tar.gz
```

This will unpack files into the `packages/thinlinc` directory, as well as the file `README.thinlinc` which contains some summarized information on the package.

Edit the `build.conf` and add a line 'package thinlinc' in the Applications section.

Run the `build` script and wait for its completion.

If everything went well, there will now be Thinstation images available in the `boot-images` directory. Use the appropriate boot image for your preferred boot method.

### 8.6.2. Configuring the ThinLinc client when running on a Thinstation Terminal

When running on a network-booted Thinstation terminal, the client is configured by adding statements to the configuration file that is downloaded at boot by Thinstation. The default name of this file is `thinstation.conf.network`, located in your `tftpboot`. There can also be other filenames that configures specific terminals based on their IP or hardware (MAC) addresses.

### 8.6.2.1. Basic Configuration

For the ThinLinc client to appear at all, a Thinstation "session" must be created. This is done by adding a few lines to the `thinstation.conf.network` file. Here's an example:

```
SESSION_0_TYPE=thinlinc
SESSION_0_THINLINC_SERVER=demo.thinlinc.com
SESSION_0_THINLINC_OPTIONS="-u user -c"
SESSION_0_THINLINC_CONFIG_NFS_SERVER_ENABLED=0
```

The above example will make the ThinLinc appear on the display of the client after boot. It will set the servername to *demo.thinlinc.com*, and it will reset the username field and enable the control panel. It will also disable export of local drives. See below for information on enabling local drives on Thinstation.

All standard client options can be added to the `SESSION_0_THINLINC_OPTIONS` variable. For example, to lock down the server field, add *-l server*.

### 8.6.2.2. Configuration using the Client Configuration File

Some of the features of the ThinLinc client can't be configured via command line options. Instead, the configuration file must be altered. To allow features such as local drive and sound redirection to work when running on Thinstation, the ThinLinc client package for Thinstation has features for altering the configuration file on the client.

To alter the configuration file, add statements on the form

`SESSION_0_THINLINC_CONFIG_<configuration file variable name> = <value>` to `thinstation.conf.network`. An example follows:

```
SESSION_0_THINLINC_CONFIG_NFS_SERVER_ENABLED=1
SESSION_0_THINLINC_CONFIG_SOUND_ENABLED=1
```

The above example will set the `NFS_SERVER_ENABLED` to *1* and the `CONFIG_SOUND_ENABLED` to *1*, and so on.

### 8.6.2.3. Enabling Sound and Local Drive Redirection

If the hardware running Thinstation has support for it and the correct sound and disk device modules have been loaded, the ThinLinc client will be able to support sound and local drive redirection. The following configuration lines in `thinstation.conf.network` will enable sound redirection and local drive redirection for USB storage devices:

```
SESSION_0_THINLINC_CONFIG_NFS_EXPORTS=/mnt/usbdevice,rw,/mnt/cdrom,ro
SESSION_0_THINLINC_CONFIG_NFS_SERVER_ENABLED=1
SESSION_0_THINLINC_CONFIG_SOUND_ENABLED=1
SESSION_0_THINLINC_CONFIG_NFS_ROOT_WARNING=0
```

### 8.6.2.4. Avoiding Question about Server Host Key

When running on a device with non-volatile storage, such as a hard disk, the ThinLinc client stores the public part of the ssh host key of the ThinLinc client the first time it connects to the server after asking

the user to verify the fingerprint of the key. At subsequent connects, this copy is used to verify that the client is connecting to the correct server.

When running on a diskless Thinstation host, the key can be stored only in volatile memory (on a RAM disk), so the client will ask the user to verify the fingerprint once each time the client has been rebooted. Since its normal behaviour to reboot a Thinstation terminal once a day, this will lead to a confusing situation for users, not to mention that it will decrease security.

To solve this problem, the ThinLinc client package for Thinstation tries to download a file name `ssh_known_hosts` from the tftproot. If it exists it will be used as database of known hostkeys on the client.

To create this file, login with the client to the ThinLinc server, using the same servername as the one that will be configured on the clients. Then copy the file `~/.thinlinc/known_hosts` to `<tftproot>/ssh_known_hosts`.

## 8.7. Web Integration and HTML5 Browser Client

This section includes information about the ThinLinc client types that can be used in conjunction with a Web Browser. The HTML5 client, and how to launch the Native ThinLinc client from the browser using the CGI script, are described below.

### 8.7.1. Launching the Native Client From a Web Page

It is possible to launch the native ThinLinc client from a web page. The process works like this:

1. The CGI script is called with the desired parameters.
2. (optional) A web page containing the Native Client Verification Applet is generated. This applet is used to verify that the native client is installed. If not, the applet allows the user to install the client.
3. The CGI script generates a "launch file", which is a normal client configuration file. When the browser receives this file, it launches the locally installed ThinLinc client.

**Note:** Only the Windows and Linux client packages configures the system to recognize launch files, and the Native Client Verification Applet can only verify if the client is installed on Windows and modern Linux systems.

**Note:** The Native Client Verification applet requires Java support in the browser. Java 7 contains a bug: It fails to retrieve the applet if the client's Server Name Indicator specified a hostname not supported by the server. To work around this issue, make sure your web server accepts all names in use. For example, with Apache, you can specify a `ServerName` and/or `ServerAlias` directives. The server can be configured to accept all names by specifying:

```
<VirtualHost _default_:443>
    ServerName catchall.example.com
    ServerAlias *
    ...
```

The launch file delivered to the client is generated from the template `/opt/thinlinc/etc/tlclient.conf.webtemplate`. The CGI script performs some substitutions on this file, before sending it to the client. Currently, the following variables are substituted:

`$server_name$`

The server name where the CGI script resides.

`$login_name$`

The user name, specified by the `username` CGI parameter.

`$password$`

The password in hexadecimal ASCII notation, specified by the `password` or `hexpassword` CGI parameters.

`$autologin$`

The value of the `autologin` CGI parameter.

### 8.7.2. The CGI Script `tlclient.cgi`

The CGI script `tlclient.cgi` is used to start the native client, when launched from a web page. It accepts many parameters which affects its operation. These are described below:

`server_name`

The desired server name.

`username`

The desired user name. No default.

`password`

The desired password, in plain text. No default.

`hexpassword`

The desired password, in hexadecimal ASCII notation. This parameter overrides the `password` parameter. No default.

`defaultinstaller`

Used by the Native Client Verification Applet. Specifies the URL to redirect to when the user initiates a client installation, on operating systems other than Linux and Windows. Default value: `http://www.cendio.com/downloads/clients`

`linuxinstaller`

Used by the Native Client Verification Applet. Specifies the URL to redirect to when the user initiates a client installation, on Linux. Default value:  
<https://www.cendio.com/downloads/clients/thinlinc-client-latest.i686.rpm>

`windowsinstaller`

Used by the Native Client Verification Applet. Specifies the URL to redirect to when the user initiates a client installation, on Windows. Default value:  
<https://www.cendio.com/downloads/clients/tl-latest-client-windows.exe>

`verify`

A boolean value which specifies if the Native Client Verification Applet should be used. Only relevant for the native client type. Default value: 1

`redirto`

After launching the native client, the browser will redirect to the web page specified by this parameter. Default value: the empty string.

`loginsubmit`

This boolean parameter specifies if a login should be directly executed, instead of showing a login form. Default value: 0

`autologin`

This boolean parameter specifies if the native client should automatically connect to the specified server at startup. Default value: 1

`start_program_enabled`

This boolean parameter specifies if the native client should request that the server starts the session with the command supplied by the client, as indicated by the `start_program_command` parameter. Default value: 0.

`start_program_command`

This parameter specifies the command to use when starting the session. Default value:  
"tl-single-app firefox".

`displayurl`

This boolean parameter can be used for debugging and development purposes. It will display and URL with all submitted parameters, and do nothing else. Default value: 0

`shadowing_enabled`

This boolean parameter specifies if the native client should activate shadowing. Default value: 0

`shadow_name`

This parameter specifies the user to shadow. Default value is the empty string.



To make it easier to test various parameters, the HTML file `cgitest.html` is included, in the same location as `tlclient.cgi`. It also demonstrates how to create icons on web pages, which launches ThinLinc sessions.

### 8.7.3. ThinLinc Web Access (HTML5 Client)

The HTML5 client is based on the Open Source project noVNC (<http://kanaka.github.io/noVNC/>). It uses HTML5 features such as WebSockets and Canvas. Since this technology is new, only a limited set of browsers are expected to work. We have tested the latest versions of following browsers:

- Internet Explorer
- Firefox
- Google Chrome
- Safari

**Note:** At the time of writing, the "Android Browser" does not work since it does not support WebSockets. For more information, see <http://code.google.com/p/android/issues/detail?id=25221>.

#### 8.7.3.1. Server Side

The server side components for the HTML5 Client are included in the package `thinlinc-webaccess`. It is automatically installed when the ThinLinc software is installed. The default TCP port number for this HTTP service is 300. It can be changed to some other port such as 443 (see below), assuming this port is free. The configured port must be allowed in any firewalls. In a cluster setup, the `tlwebaccess` service must run on all machines. The same service port should be used, and all machines must be accessible from the clients. Also, observe the following limitations:

- For best security and user experience, we recommend that you use valid TLS certificates. The certificates should match the server host names.
- With some browsers such as Safari on iOS, the certificates MUST match the server hostname. This means that you cannot connect through an IP address. The bundled certificate is created for "localhost" and cannot be used. You can create a new certificate using our shipped helper script `make-dummy-cert` as:

```
$ sudo /opt/thinlinc/etc/tlwebaccess/make-dummy-cert `hostname --fqdn`
```

Additionally, a self signed certificate must be manually approved on the iOS device. This can be done by clicking on the link "Download Server Certificate" in the web interface. This must be done for all machines in a cluster.

**Note:** OpenSSL is required to be installed to use the shipped helper script `make-dummy-cert`.

**Note:** Google Chrome does not work with self-signed certificates on iOS, a trusted root certificate is required. For more information, see <https://code.google.com/p/chromium/issues/detail?id=152584>.

#### 8.7.3.1.1. Configuration

tlwebaccess offers a number of configuration options, which are stored in the configuration file `/opt/thinlinc/etc/conf.d/webaccess.hconf`. The various configuration options are described below.

`/webaccess/cert`

The path to the certificate file to be used for TLS encryption.

**Note:** This certificate may be downloaded by connecting clients to be installed in their browsers. Make absolutely sure that this file does not contain a private key.

`/webaccess/certkey`

The path to the certificate private key file used for TLS encryption.

`/webaccess/login_page`

The redirection URL to the login page on the master machine. The default value, which is empty, will not redirect to another server and is only usable if you are running a stand alone ThinLinc server. If you have more than one server or is using Network Address Translation (NAT), you must set this parameter to an address that all clients can connect to. Example:

```
login_page = https://thinlinc.example.com:300/
```

`/webaccess/listen_port`

The local port for this service to listen on. The default port used is 300.

`/webaccess/logging/logfile`

The file to use for logging tlwebaccess messages. By default, this is `/var/log/tlwebaccess.log`.

#### 8.7.3.2. Browser Side

The HTML5 client is accessed with your web browser by browsing to the master machine, for example `https://thinlinc-master.example.com:300`. If you have configured the service to run on port 443, `":300"` can be omitted.

**Note:** On iOS devices, you can add an icon to the home screen which is called web clip. When the ThinLinc Web Access is launched from the home screen, it will run in full screen mode.





## Chapter 9. Authentication in ThinLinc

In this chapter we will describe how authentication of users is performed in ThinLinc

### 9.1. Pluggable Authentication Modules

Authentication of users in ThinLinc is performed using the *Pluggable Authentication Modules* (PAM). This means ThinLinc can authenticate users using any system for which there is a PAM module. Examples of PAM modules are *pam\_ldap* for accessing LDAP directories (including Novell NDS/eDirectory) and *pam\_winbind* for authenticating against a Windows Domain. Of course, authentication using the standard plaintext password files of Linux is also possible using the PAM module *pam\_unix*.

If ThinLinc should authenticate against the `passwd` database on the local host, no configuration at all is needed, since this is how most distributions are configured at installation. However, at many sites there is already some type of existing user database. In this chapter we'll go into detail on how to authenticate ThinLinc users against Windows domains and LDAP databases.

An user connecting to ThinLinc needs executable access to the ThinLinc login shell *thinlinc-login* and if you don't have any intentions to allow a regular shell access to the server you should set default login shell for the users to `/usr/bin/thinlinc-login`.

#### 9.1.1. Configuration files for PAM

PAM is configured by editing the files located in the directory `/etc/pam.d/` (at least in the distributions we've tested ThinLinc on).

Different Linux distributions have slightly different ways of configuring PAM. The ThinLinc installation program will setup ThinLinc to authenticate using the same PAM setup as the Secure Shell Daemon, by creating a symbolic link from `/etc/pam.d/thinlinc` to either `/etc/pam.d/sshd` or `/etc/pam.d/ssh`, depending on which of the latter files that exists at installation. This seems to work on most distributions. Be aware that the PAM settings for the Secure Shell Daemon might really be somewhere else. For example, on Red Hat distributions, the file `/etc/pam.d/system-auth` is included by all other pam-files, so in most cases, that is the file that should be modified instead of the file used by `sshd`.

### 9.2. Limitations

Some PAM modules and authentication mechanisms are case sensitive, while others are not. Usernames in the ThinLinc client are case sensitive by default, however this behaviour can be changed. See `LOWERCASE_LOGIN_NAME` in Section 7.7 for details.

## 9.3. Using Novell eDirectory with ThinLinc

### 9.3.1. Configuring eDirectory and ThinLinc with TLNC

Within ThinLinc, a tool called ThinLinc Novell Configurator (TLNC) helps in the task of configuring ThinLinc to interoperate with a Novell eDirectory server. By using this tool, most of the tasks needed to configure the eDirectory and ThinLinc servers are automated.

The Novell Configurator is available as a module in the ThinLinc Web Administration as documented in Chapter 16.

#### 9.3.1.1. Tasks Performed by TLNC

The ThinLinc Novell Configurator takes care of the following tasks:

- *Verifying existence of the `posixAccount` and `posixGroup` objectclasses*

Some older versions of eDirectory lacks the relevant LDAP schema needed for proper operation with a ThinLinc server. If they don't exist, they can most often be added by installing the Native File Access for Unix product from Novell.

- *Check for and remove incorrect attribute mappings*

Some older versions of eDirectory, among them eDirectory on Netware 6.0, has incorrect attribute mappings, mapping the NDS attribute UID to `uidNumber`, and GID to `gidNumber`. If they exist, eDirectory will not function properly with a ThinLinc server.

- *Create user object needed for search operations*

The ThinLinc servers need to run LDAP search operations, and to be able to access the relevant attributes, they will bind as a special user object with DN and password. The TLNC can create this user, if it doesn't exist.

- *Give relevant permissions to the search user object*

The user object used for search operations need to have access to a number of attributes. The TLNC can modify the ACL (Access Control Lists) of the eDirectory server to allow the search user object to read the relevant attributes. TLNC will also create the file

`/opt/thinlinc/etc/ldap.conf.template` that can be used as a template for the configuration file needed for `pam_ldap` and `nss_ldap`.

- *Create user object needed for `tl-nds-posixuser/tl-nds-posixgroup`*

In most cases where ThinLinc is integrated with a eDirectory server, the **tl-nds-posixuser** and **tl-nds-posixgroup** tools are used to add attributes to existing user and group objects in the directory to make them usable in the ThinLinc environment. The TLNC can create the user object needed by the tools to access eDirectory. It will also write information about this user to the relevant configuration file.

See Section 9.3.4 for detailed information on **tl-nds-posixuser** and **tl-nds-posixgroup**.

- *Give relevant permissions to user object used by `tl-nds-posixuser/tl-nds-posixgroup`*

The user object needed by **tl-nds-posixuser** and **tl-nds-posixgroup** need to have read and write access to a number of attributes. The TLNC can modify the ACL (Access Control Lists) of the eDirectory server to allow the user object to read the relevant attributes.

- *Create a default group to be used for ThinLinc Users*

All users of a ThinLinc server need to have a default posixGroup assigned to their user objects. This group can be created by TLNC.

- *Create indices needed for proper performance*

For proper performance, a few indices on LDAP attributes are needed. The Novell Configurator can create the needed indices.

### 9.3.1.2. Using the ThinLinc Novell Configurator

TLNC is normally used only a few times - at installation, and when new eDirectory servers are added to the network. It can be run at any time to verify that all settings are correct. It is recommended to run TLNC after each upgrade of ThinLinc, so new recommended settings in eDirectory can be applied. Also run TLNC when new eDirectory servers are added, to correctly configure attribute mappings and indices.

To run TLNC, point your web browser to your ThinLinc server, port 1010, as documented in Chapter 16. Look for the menu with the text *Novell Configurator* and click on it. You have now entered the TLNC.

Select the *Configure* submenu, and start by entering values into the fields provided by the web page you just entered.

- The *eDirectory Server Hostname* should be the fully qualified hostname of one of your eDirectory servers. It doesn't matter which server you choose, TLNC will read out the list of LDAP servers from the first server, and configure all of them.
- Activate the *Connect using SSL* checkbox to communicate encrypted with the eDirectory server.

**Note:** You *must* use the fully qualified hostname, i.e., a hostname on the form *server.example.com* in the *eDirectory Server Hostname* field, or SSL connections will fail.

- Normally, you don't have to enter anything into the *Search base* field, since most eDirectory sites have a structure that enables TLNC to work well when searching from "". For sites that have several LDAP trees below the root, this setting may be of need.
- Enter the DN of the administrative user into the *Novell Admin User DN* field. The DN should be entered in LDAP-form, with comma signs between the different parts of the DN, not periods.
- Finally, enter the password of the administrative user into the *Password of Admin User* field.

Now press the **Start Configuration** button. The Novell Configurator will now communicate with your eDirectory server to see what needs to be done. It will then present a page where you can see the status of the integration between the ThinLinc cluster and eDirectory. Areas where action is needed are marked with red.

Inspect the data presented on the page. In the areas where action is needed, TLNC will have activated the checkboxes for the action that needs to be taken. Verify that all checked actions are things you want, and then press the **Execute selected actions** button. The TLNC will now communicate further with your eDirectory server to execute the actions you selected.

When all actions are executed, the same page will show up again, with the integration status. Verify that all went well, and proceed with the actions needed after running TLNC.

### 9.3.1.3. Proceeding with eDirectory Integration after running TLNC

After running TLNC, a few things need to be done manually

#### 9.3.1.3.1. Test and Distribute `ldap.conf`

After creating the search user needed by `pam_ldap` and `nss_ldap` in LDAP, TLNC writes the file `/opt/thinlinc/etc/ldap.conf.template` with information about the DN and password of the created user. The file resides on the filesystem of the server that hosts the ThinLinc Web Administration service.

This file can serve as a good starting-point for the contents of the real configuration file for `pam_ldap` and `nss_ldap`. The recommended procedure is to configure LDAP authentication using your distribution's tools, and then replace the `ldap.conf` generated with the one generated by TLNC.

#### 9.3.1.3.2. Configure Missing Index and Default POSIX Group if Needed

Some versions of eDirectory fail to create an index on the Object Class attribute via LDAP. The reason for this is unknown, but if TLNC indicates that this is the case, the index needs to be configured by hand. See Section C.2 for documentation on how to do this manually.

Also, some versions of eDirectory fail to create the default POSIX group `tl-users` via LDAP. Reason for this is also unknown, but if TLNC indicates that this is the case, create the default group by hand by creating a group object with `cn=tl-users` in eDirectory, and assign a `gidNumber` of 1000 to it.

#### 9.3.1.3.3. Trigger a LIMBER Run on the eDirectory Servers

When an index is created in eDirectory via LDAP, for example by TLNC, they are not immediately built and used. Instead, eDirectory waits for a process called LIMBER to run. There are several ways to make this happen:

- *Wait*

LIMBER is automatically run every three hours, or when another server starts a connectivity check. When this happens, all indexes pending creation will be built.

- *Trigger a run via `DSTRACE/ndstrace`*

On Netware, load `DSTRACE`, and then run **SET DSTRACE=\*L**. This will trigger LIMBER to run immediately.

If the eDirectory server runs on Linux, for example on Open Enterprise Server, a LIMBER run can be triggered by running **ndstrace**. Within `ndstrace`, write **set dstrace=\*l** and press enter. This will also trigger a LIMBER run.

- *Trigger a run via `iManager`*

Enter iManager, Agent Configuration/Agent triggers, and trigger a LIMBER run.

After triggering the LIMBER on all eDirectory servers, run TLNC again to see if the status of the defined indices have changed to online.



#### 9.3.1.3.4. Run **tl-nds-posixuser** and **tl-nds-posixgroup**

Run **tl-nds-posixuser** and **tl-nds-posixgroup** as described in Section 9.3.4 to complement your existing users and groups with values that allow them to be used with the ThinLinc server(s).

### 9.3.2. Acquiring the SSL CA Certificate from eDirectory

In order to use SSL with server verification, the public part of the CA Certificate used to sign the certificate used for SSL on the LDAP server in eDirectory must be extracted. This is done using the following procedure:

1. Find out what SSL Certificate object is used for the LDAP server in question using eDirectory. Find the LDAP Server object, choose properties and in the "SSL Configuration" tab, look up the name of the SSL Certificate object in use.
2. Find the Certificate Object, choose properties and in the "Certificates" tab, choose "Trusted Root Certificate". Press the "Export" button. If available, choose "File in base64 format", if not, choose "File in binary DER format".
3. Choose an appropriate filename and press the Export button. A file with the certificate will now be written to your harddisk.
4. If you managed to export the file in base64 format (which is the same thing as PEM), use the file as `tls_cacertfile`. If the export was done to DER format, execute the following command to convert it to PEM:

```
[root@test root]
      openssl x509 -inform der -in
      <filename.der> -outform pem -out
      <filename.pem>
```

Use the resulting file as `tls_cacertfile`

### 9.3.3. Allowing Clear Text Passwords (bind operations)

If there by some reason is no way to enable transport security (SSL) for bind operations (where passwords are transmitted), eDirectory must be configured to allow cleartext bind operations. This is done in the LDAP Group object under the General tab.

### 9.3.4. Using eDirectory User and Group Objects with ThinLinc

When integrating a Linux server into a Novell Netware network with eDirectory as source for authentication information for the first time, the users in the eDirectory most probably doesn't have the necessary objectclasses and attributes defined. The users must have the objectclass `posixAccount` and the attributes (from `posixAccount`) `uidNumber`, `gidNumber`, `loginShell`, `uid`, and `homeDirectory` defined. Groups must have the objectclass `posixGroup`, and a `gidNumber` defined. Manually assigning the required values is not practical for installations with more than a few users.

In ThinLinc, this problem is solved using two programs, **tl-nds-posixuser** and **tl-nds-posixgroup** that searches the directory for users that don't have `posixAccount` among their objectclasses, and for groups that don't have `posixGroup` among their objectclasses. When it finds such a user or group, the values required are automatically assigned. Optionally, **tl-nds-posixuser** can be used to assign filesystem permissions to home directories exported from Novell Netware servers in Independent mode (see Section 10.2).

The idea is that **tl-nds-posixuser** and **tl-nds-posixgroup** are first run during deployment of the ThinLinc installation. At this occasion, all existing users and groups are assigned the `posixAccount` attributes. After deployment, **tl-nds-posixuser** and **tl-nds-posixgroup** are meant to be run from cron at regular intervals, for example every 15 minutes. This way, new users and groups added to the directory will be assigned the relevant attributes automatically. This means that new users will be able to log in to the ThinLinc servers without the administrator having to take any action, after a short delay determined by the interval between each **tl-nds-posixuser** invocation.

#### 9.3.4.1. Configuration of tl-nds-posixuser and tl-nds-posixgroup

**tl-nds-posixuser** and **tl-nds-posixgroup** both need a set of configuration parameters to be able to perform their duties. Some of the parameters are common to the two programs, others are command-specific. Parameters common to the two are stored under the `/utils/tl-nds/` `hiveconf` folder. Parameters specific to **tl-nds-posixuser** are stored under the `/utils/tl-nds/posixuser/` `hiveconf` folder, and parameters specific to **tl-nds-posixgroup** are stored under the `/utils/tl-nds/posixgroup/` `hiveconf` folder.

It is a good idea to run the ThinLinc Novell Configurator as documented in Section 9.3.1. This creates the user object needed by **tl-nds-posixuser** and **tl-nds-posixgroup**, and writes down information about the server, user DN and password to `/utils/tl-nds/`.

All parameters specified in `hiveconf` can be overridden using command line parameters.

**Table 9-1. Configuration Parameters Common to Both tl-nds-posixuser and tl-nds-posixgroup**

Hiveconf parameter path	Command line parameter	Explanation
<code>/utils/tl-nds/ldapuri</code>	<code>--ldapuri</code>	The LDAP URI to connect to. This may be either a <code>ldap://</code> or <code>ldaps://</code> , the latter for LDAP over SSL.
<code>/utils/tl-nds/binddn</code>	<code>--binddn</code>	The DN of the object that should be used to bind to the eDirectory. This DN must have permissions to change some attributes of all relevant users in the directory. See below for information on the exact procedure for creating this object.

Hiveconf parameter path	Command line parameter	Explanation
<code>/utils/tl-nds/bindpw</code>	<code>--bindpw</code>	The password of the <code>binddn</code> . As noted above, the big advantage of using a hive for storing this is that the hive can be protected so that only a specific user may see the contents of the specific part of the hive. This way, the password is not exposed to all users on the host running <b>tl-nds-posixpath</b> .
<code>/utils/tl-nds/searchbase</code>	<code>--searchbase</code>	The point in the tree from where searches should begin when trying to find users and groups without <code>posixAccount/posixGroup</code> attributes set.
	<code>--verbose</code>	Print out what happens during execution.
	<code>--simulate</code>	Don't do anything, just simulate what would have been done. Only useful in combination with <code>--verbose</code>

**Table 9-2. Configuration Parameters Specific to tl-nds-posixuser**

Hiveconf parameter path	Command line parameter	Explanation
<code>/utils/tl-nds/posixuser/cachefilename</code>	<code>--cachefilename</code>	The file name of the filename where the last used uidNumber is stored. The default is <code>/var/opt/thinlinc/utils/tl-nds/posixuser/cachefilename</code> and generally, there's no reason to change this value. This file is locked by a running <code>tl-nds-posixuser</code> process to protect against duplicate uidNumber values in LDAP.

Hiveconf parameter path	Command line parameter	Explanation
/utils/tl-nds/posixuser/exclude-dn	--exclude-dn	A list of entries in the LDAP tree to ignore. If setting this in hiveconf, use a space-separated list of DN's. If setting on the command line, give several --exclude-dn with one DN after each. This setting can be used if there are places in your LDAP tree where for example invalid users are stored. The DN itself and all its leaves will be ignored.
/utils/tl-nds/posixuser/ldapfilter	-l	The LDAP filter (as defined in RFC2254) to be used for finding the users that should be modified. Default value is (&(object-class=inetOrgPerson)(!(objectclass=posixAccount)),
/utils/tl-nds/posixuser/groupid	-g	The group that should be set as primary group for all users. This should be the group id (A number) of the group, not the name.
/utils/tl-nds/posixuser/loginshell	-s	Login shell set on users (in the loginShell) attribute. The default is /bin/bash
/utils/tl-nds/posixuser/update-script	-u	This parameter points to a executable script that is run just before the the user object in eDirectory is updated. The script gets three parameters - the username, the uidNumber and the gidNumber. Example usage of this script is creation of home directories or permission changes on existing home direcotires.
/utils/tl-nds/posixuser/uidfield	-i	This parameter can be used to decide what attribute of a user object in LDAP should be used as username. The default value is 'cn' which is OK for most eDirectory installations.

Table 9-3. Configuration Parameters Specific to tl-nds-posixgroup

Hiveconf parameter path	Command line parameter	Explanation
-------------------------	------------------------	-------------

Hiveconf parameter path	Command line parameter	Explanation
<code>/utils/tl-nds/posixgroup/cachefilename</code>	<code>--cachefilename</code>	The file name of the filename where the last used gidNumber is stored. The default is <code>/var/opt/thinlinc/utils/tl-nds/posixgroup/cachefilename</code> and generally, there's no reason to change this value. This file is locked by a running <code>tl-nds-posixgroup</code> process to protect against duplicate gidNumber values in LDAP.
<code>/utils/tl-nds/posixgroup/exclude_dns</code>	<code>--exclude-dn</code>	A list of entries in the LDAP tree to ignore. If setting this in hiveconf, use a space-separated list of DNs. If setting on the command line, give several <code>--exclude-dn</code> with one DN after each. This setting can be used if there are places in your LDAP tree where for example invalid groups are stored. The DN itself and all its leaves will be ignored.
<code>/utils/tl-nds/posixgroup/ldapfilter</code>	<code>--ldapfilter</code>	The LDAP filter (as defined in RFC2254) to be used for finding the groups that should be modified. Default value is <code>(&amp;(objectclass=groupOfNames)(!(objectclass=posixGroup)))</code>

### 9.3.5. Forcing Users to Change Passwords in an eDirectory Environment

In most environments, forcing the users to regularly change password is a good security measure. This section will describe how to do this with ThinLinc in an eDirectory environment.

#### 9.3.5.1. Functionality

By activating forced password changes in ThinLinc, a graphical window will popup during the login sequence if the password expire date is near, or if the password has expired. There is also support for the "grace logins" functionality of eDirectory. The functionality can be summarized as below:

- At login, **tl-nds-check-expired** sends a query to the LDAP server defined in `/etc/ldap.conf` and asks for information about password expiration.
- If the number of days until the password expires is less than `warning_days` (see below), **tl-nds-check-expired** pops up a window and warns the user that the password is about to expire, and

asks if the password should be changed now, or not.

- If the password has expired, and "grace logins" are enabled, **tl-nds-check-expired** will pop up a window and tell the user how many logins he/she has left before the account will be locked. The user will again be asked if he/she wants to change the password.
- If the number of "grace logins" left is less than 3, the pop up window will inform the user that the current login attempt is the last login attempt before the account will be locked, and ask if the user wants to change password.
- If the account is locked, the ThinLinc client will say that the username or password was invalid. No session will be created.

### 9.3.5.2. Configuration

Before trying to use **tl-nds-check-expired**, make sure that **tl-passwd** works, since it's used by **tl-nds-check-expired**. Pay special attention to the note about `pam_password` in the part about the **tl-passwd** in Chapter 13 to make sure it's set to a value appropriate for the eDirectory environment.

**tl-nds-check-expired** has one parameter, `/utils/tl-nds-check-expired/warning_days`, in the file `tl-nds-check-expired.hconf` located among the rest of the ThinLinc configuration files on the server. The `warning_days` parameter decides how many days before the password for a specific user expires that **tl-nds-check-expired** begins to warn about the expire date. If set to 0, it will not warn until the password actually has expired.

### 9.3.5.3. How to Activate Force Password Changes

Start by verifying that the search DN created and configured either via the ThinLinc Novell Configurator (TLNC) (see Section 9.3.1) or via the procedure described in Section C.5 has access to the attributes `passwordExpirationTime`, `loginGraceRemaining`, and `loginGraceLimit`. Then create a symbolic link in `/opt/thinlinc/etc/xstartup.d` pointing at the **tl-nds-check-expired** command:

```
ln -s /opt/thinlinc/bin/tl-nds-check-expired /opt/thinlinc/etc/xstartup.d/08-tl-nds-check-e
```

Now continue by testing to login as a user with a password that will soon expire, or a password that has expired and has grace login attempts left.

### 9.3.5.4. Interaction with Application Servers and Fat Clients

**IMPORTANT:** Due to the internal structure of ThinLinc, each time a user logs in, two separate authentications are performed. This means that if "grace logins" are enabled, each time a user logs in after the password expiration time, two grace login attempts are consumed. The number of grace logins available should be adjusted to compensate for this. Grace login attempts are also consumed at session startup, if any NCP resources are mounted.

Also, if a user uses both ThinLinc and fat workstations, another problem might occur. If the user has logged in to fat workstations and ignored the prompt for password change so that the number of grace logins left is 1 or 2, the user will be locked out the next time he/she tries to login to ThinLinc, since the ThinLinc session will only be setup halfway before the server refuses to authenticate the user.

Further problems occur if the ThinLinc cluster is combined with Application Servers, for example Windows Terminal Servers, to provide access to Windows Applications from ThinLinc. If "grace logins" are enabled, each time the user starts an application on the Application server, one grace login attempt will be consumed. One way of limiting this problem is to check if the password is about to expire in the Netware login script. This can be done using the `PASSWORD_EXPIRES` or by using third-party software such as **passXchk** available at <http://www.novell.com/coolsolutions/tools/1911.html>. If the password change popup shouldn't occur if the Windows Terminal Server session is used to execute a single program, the program **tl-is-appsession** can be used to determine if the session is an application session or not, in the login script.

### 9.3.6. Configuring Windows Terminal Servers with Netware Client for Single Sign-On

In environments where ThinLinc users should be able to access software running on Windows Terminal Servers, with ThinLinc Single Sign-On support, some extra configuration is needed if the Windows Terminal Servers are configured to logon to a Novell network. The extra configuration is needed to make sure the user can automatically login without having to manually enter the context where his/her user object is located.

To solve this problem, the username is sent *with* context. For example, for a user with `cn=user5` located in the ou `ou=school1,o=organization`, the username will be sent as `cn=user5.ou=school1.o=organization` over RDP.

For this to work, the Novell client installed on the Windows Terminal Server must be recent enough to support receiving usernames with context over RDP. Version 4.91 SP4 is known to work.

Activate login to Windows Terminal Servers with context by adding the script **tl-set-novelluser.sh** to `/opt/thinlinc/etc/xstartup.d`:

```
ln -s /opt/thinlinc/libexec/tl-set-novelluser.sh /opt/thinlinc/etc/xstartup.d/07-tl-set-nov
```

This will set an environment variable at session startup. This environment variable is then detected by **tl-run-rdesktop** when run.

**Note:** Please note that for login to work, users must exist not only in eDirectory, but also in the Windows environment, for example via an AD domain or by using Zenworks dynamic local users feature. Setup of this is however outside the scope of this document.

## 9.4. Using Public Key Authentication

### 9.4.1. Introduction

Public key authentication is a more secure alternative to passwords. It uses a challenge/response

mechanism that prevents even the server from gaining access to the authentication information. This section will describe how to configure ThinLinc to use it.

### 9.4.2. Key Generation

In order to use public key authentication, a pair of encryption keys must be generated. Tools to accomplish this should be included with the SSH server. On Linux, that server is normally OpenSSH and the tool to generate keys is called **ssh-keygen**.

Example:

```
# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/test/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/test/.ssh/id_rsa.
Your public key has been saved in /home/test/.ssh/id_rsa.pub.
The key fingerprint is:
e1:87:0d:82:71:df:e9:4a:b0:a8:e3:cd:e8:79:58:32 testuser@example.com
```

Remember that the private key (`id_rsa` in the example) is a password equivalent and should be handled with care. The public key (`id_rsa.pub` in the example) does not need to be kept secret.

**Note:** The SSH key format is not standardised, so it may be required to convert the keys depending on which servers you will be using.

### 9.4.3. Server Configuration

All SSH servers must support public key authentication, so any SSH server will work. It is important, however, to verify that public key authentication is not disabled. Refer to the documentation for your SSH server for instructions on how to do this.

Next, the public keys need to be associated with the correct users. For OpenSSH, you must store a copy of the public key in the users' home directory, specifically in the file `~/.ssh/authorized_keys`. This file can contain multiple keys, each on a separate line.

### 9.4.4. Client Configuration

The client must have a copy of the private key associated with the public key stored on the server. The key can be stored anywhere, although on UNIX it is commonly stored as `~/.ssh/id_rsa`. The user will be able to specify where the key is located in the ThinLinc Client interface.



## 9.5. Using Smart Card Public Key Authentication

### 9.5.1. Introduction

Smart card public key authentication is an advanced version of the method described in Section 9.4. It uses the same basic principle but stores the private key on a smart card, where it can never be extracted. This section will describe how to configure ThinLinc to use it.

### 9.5.2. General Requirements

- Smart cards with an appropriate PKCS#11 library. The library included with ThinLinc requires PKCS#15 compliant smart cards and PC/SC libraries on the client system.

### 9.5.3. Key Generation

The keys on the smart card are generated when the smart card is issued. How this is done is not covered by this guide.

### 9.5.4. Server Configuration

To use a smart card with ThinLinc, the public key must be extracted off the card and associated with a user on the ThinLinc server. The method for doing this depends on your smart card and your SSH server.

On Linux, with the OpenSSH server and an PKCS#15 compliant smart card, the tool **pkcs15-tool** (part of the OpenSC suite) is able to extract the public key.

The first step is identifying the certificate on the card:

```
# pkcs15-tool --list-certificates
X.509 Certificate [identification]
    Flags      : 0
    Authority: no
    Path       : 3f0050154331
    ID         : 45
```

The second step is to extract the key, based on the ID number:

```
# pkcs15-tool --read-ssh-key 45
1024 65537 918282501237151981353694684191630174855276113858858644490084487922635
27407657612671471887563630990149686313179737831148878256058532261207121307761545
37226554073750496652425001832055579758510787971892507619849564722087378266977930
9875752082163453335538210518946058646748977963861144645730357512544251473818679
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCCxIx/xtVoDR2qwY4Pym7F6yKmdJsB26MUbbTiGT7o
6M6G4A2l5Go1kdQRNjOWDJE9HZWToaApSkVprNeiQLeOkbELz2yND2Te/Oyl3u44YeIUImTlv4t7q9jC
MTpfZ+TpxLf0sd3DAk2So8EBAAtUkhib/ugKqfTCqB7WN0Hf0Nw==
```

The second line, starting with "ssh-rsa", is the one needed for SSH version 2 authentication. For instructions on how to associate this key with a user, see Section 9.4.

### 9.5.5. Client Configuration

The ThinLinc client requires no special configuration to use the smart card.

### 9.5.6. Automatic Connection

The client is able to automatically connect to the server when a smart card is inserted (see Section 7.4.5). It does, however, require that the user is able to log in using the subject name on the card. As that is rarely a valid user name, ThinLinc ships with a special NSS module, called *nss-passwdaliases*, that enables alternate names for users.

The module is configured by editing the file `/etc/passwdaliases`. The file is a colon-delimited table of alternate names and their corresponding user ids. Example:

```
givenname=John,sn=Doe,c=us:572
```

To activate the *nss-passwdaliases* module, it must be added to the list of NSS modules for the *passwd* database. This is specified in the file `/etc/nsswitch.conf`. For example, replace the following line:

```
passwd: files ldap
```

with this line:

```
passwd: files ldap passwdaliases
```

### 9.5.7. LDAP Automatic Update (tl-ldap-certalias)

ThinLinc includes the tool **tl-ldap-certalias** that can automatically update the local databases needed for smart card public key authentication, provided the system uses the OpenSSH server (or any SSH server that uses a compatible format and location for authorized public keys) and standards compliant LDAP servers where users and certificates are stored.

The **tl-ldap-certalias** command can also perform validation of certificates it finds in LDAP databases. Read more about this in Section 9.5.7.3.

- On invocation, a list of all users and matching certificates is gathered. How is determined by the `certificate_user_match` configuration variable. If `allow_invalid_certificates` is no, only matching valid certificates will be gathered.
- The user's home directory, as well as the `.ssh` directory, are created if they are required and do not already exist. **tl-ldap-certalias** reuses the `/vsmagent/make_homedir_mode` configuration variable from `vsmagent` for determining the default permissions of newly created home directories.

- Any old public keys added by **tl-ldap-certalias** are removed from the `authorized_keys` file and the keys from the current set of certificates are added.
- The file `/etc/passwdaliases` is updated with a list of subject names and user id:s, to allow for login without usernames. See Section 9.5.6 for more information.

**Note:** It should be noted that any custom entries in `~/.ssh/authorized_keys` will be retained, but custom changes to `/etc/passwdaliases` will be overwritten each time **tl-ldap-certalias** is run.

After deployment, **tl-ldap-certalias** is meant to be run from cron at regular intervals, for example every 15 minutes. This makes sure that the ThinLinc system automatically keeps all user certificates updated. However, please note that if you're using certificate validation, downloading and parsing certificate revocation lists may take a long time (up to 5 minutes each). This is mitigated by caching the data from the CRL:s, but the first run and whenever the CRL needs to be updated may take a long time. Thus, if you have certificates from a lot of different certificate authorities, don't run **tl-ldap-certalias** too often.

Since the default use of this tool is to be run from cron, the default behaviour is to produce no output other than error messages. If you want more output from **tl-ldap-certalias**, see options in Section 9.5.7.1.

**Note:** The root user must be able to write to the users' home directories for **tl-ldap-certalias** to be able to update the `~/.ssh/authorized_keys` files.

### 9.5.7.1. Command line options

**tl-ldap-certalias** accepts a number of different command line options that affects how the program interacts with its environment.

`-v`  
`--verbose`

Turn on program status output to standard output. This is off by default.

`-d`  
`--debug`

Turn on extra debugging output to standard output. This is off by default.

`-s`  
`--simulate`

Dry run mode. Specifying this option tells **tl-ldap-certalias** to avoid writing any changes to disk. This is off by default.

`-h`  
`--help`

Show usage information and exit.

### 9.5.7.2. Configuration

**tl-ldap-certalias** uses the `/utils/tl-ldap-certalias` hiveconf folder for configuration purposes. On a standard ThinLinc installation, it's located in `/opt/thinlinc/etc/conf.d/tl-ldap-certalias.hconf`.

#### Configuration parameters

`/utils/tl-ldap-certalias/ldap_schema`

Specify the schema type that is used on the target LDAP server. Valid options are `rfc2307` and `AD`. `rfc2307` is default and should be used with standard LDAP servers that complies to `rfc2307`. `AD` should be used if target LDAP server is an Active Directory.

`/utils/tl-ldap-certalias/allow_invalid_certificates`

This parameter controls whether to perform validation on certificates found in the LDAP database. Possible values are `yes` and `no`.

Please note that by setting this to `yes`, you will allow users with expired, invalid, revoked or untrusted certificates to log in as if their certificates are valid.

**Note:** If you want **tl-ldap-certalias** to match the behaviour of **tl-ldap-certalias** versions earlier than 3.2.0, set this to `yes`.

`/utils/tl-ldap-certalias/certificate_user_match`

The method to use for finding certificates assigned to the user. Valid options are `sameobject` and `novell_certificate_subjectname`.

`sameobject` makes **tl-ldap-certalias** search for certificates in the `userCertificate` attribute on user objects it finds.

`novell_certificate_subjectname` allows for certificates to be stored in another LDAP tree. User objects are associated to certificates by storing subject names of certificates in a multivalued attribute called `sasAllowableSubjectName` (OID 2.16.840.1.113719.1.39.42.1.0.69) on the user object. **tl-ldap-certalias** can handle both DN:s as written by Novell iManager (`DC=com.DC=example.OU=test.CN=foo`) and as returned by **tl-certtool --subject** (`cn=foo,ou=test,dc=example,dc=com`).

`/utils/tl-ldap-certalias/users/uri`

A LDAP server URI for finding users on the form `ldap [s] :// name [:port]`

`/utils/tl-ldap-certalias/users/base`

The LDAP search base for finding users.

`/utils/tl-ldap-certalias/users/binddn`

The username to bind as for searching for users. If left blank, no bind is performed.

```
/utils/tl-ldap-certalias/users/bindpw
```

The password to use in combination with binddn for bind operations. If binddn is left empty, this can also be left empty.

```
/utils/tl-ldap-certalias/certs/uri
/utils/tl-ldap-certalias/certs/base
/utils/tl-ldap-certalias/certs/binddn
/utils/tl-ldap-certalias/certs/bindpw
```

If certificate\_user\_match is not sameobject, these settings will be used to determine where to look for certificates. They follow the same rules as the settings for users.

### 9.5.7.3. Certificate validation

**tl-ldap-certalias** can perform validation of certificates found in LDAP databases by the following methods if allow\_invalid\_certificates is set to yes:

Certificate validity and expiry dates

**tl-ldap-certalias** now checks the certificate validity and expiry dates and rejects certificates that are not valid yet or have expired.

Matching certificate to certificate issuers

Place the CA certificates you wish to trust certificates from in `/opt/thinlinc/etc/ca/`. The CA certificates must be in DER form. If **tl-ldap-certalias** finds a certificate with an issuer that does not match any of the certificates in `/opt/thinlinc/etc/ca/`, the certificate will be considered invalid and ignored.

Certificate revocation lists

**tl-ldap-certalias** searches the certificates it encounter for certificate revocation lists (CRL), to make sure that the certificate has not been revoked by its issuer. Once downloaded, the CRL will be cached until the time for the next scheduled update found in the CRL list has passed.

**Note:** **tl-ldap-certalias** can only handle CRL lists distributed with HTTP.

Validation of certificate signatures.

**tl-ldap-certalias** can verify that the certificate signature is valid and thus assures that the certificate has not been tampered with.

**Note:** To validate rsa-sha256, rsa-sha384 and rsa-sha512 certificate signatures, Python 2.5 or newer is required. Trying to validate signatures with an older Python version will result in the certificate being rejected with the message "Certificate signature algorithm is unknown".

## 9.6. Using One Time Passwords

### 9.6.1. Introduction

In this section, we will describe how to configure ThinLinc for authentication against One Time Password (OTP) servers, such as the NordicEdge One Time Password Server or RSA SecurID. By using OTPs, you can increase the system security.

### 9.6.2. General Requirements

- An OTP server which accepts the OTP twice. This is due to the ThinLinc architecture: The client first contacts the master machine, and then the agent host. The NordicEdge One Time Password Server has built-in support for ThinLinc. When using RSA SecurID, we recommend using the Steel-Belted Radius server as a "Token Caching Server".
- An user database (directory server) that is supported both by the operating system on the ThinLinc servers, as well as the OTP server. We recommend using a LDAP directory server, such as Novell eDirectory.
- The operating systems on the ThinLinc servers must support the OTP servers authentication protocol. We recommend using the RADIUS protocol, by using the `pam_radius_auth` PAM module from the FreeRADIUS project (<http://www.freeradius.org>).
- The SSH server on the ThinLinc servers must accept "keyboard-interactive" authentication. It's recommended to disable "password" authentication.

### 9.6.3. Configuration for NordicEdge One Time Password Server

This section describes how to deploy a OTP solution based on the NordicEdge One Time Password Server. With this solution, OTPs are used in addition to normal passwords. Users should enter their normal password in the ThinLinc client password input field, and the OTPs in the popup dialog that follows.

The example below assumes that you are using LDAP and RADIUS. Step 2 through 8 should be repeated on all ThinLinc servers.

1. Configure the OTP server, according to the OTP server documentation. Make sure to use an appropriate "regenerate timeout" value. If this value is too short, authentication with the VSM agent host might fail, even though authentication with the VSM server host succeeds.
2. Install `pam_radius_auth`. Some distributions, such as SUSE, includes this module.
3. Configure `pam_radius_auth`, by creating `/etc/raddb/server` . It should contain the OTP server name, port, and a shared secret. Example:  

```
myotpserver.example.com:1645 mysecret
```
4. Configure LDAP authentication according to your distribution's documentation.

5. Normally, when not using OTPs, the VSM and SSH Server PAM configuration is the same. This is often accomplished by a symbolic link `/etc/pam.d/thinlinc` pointing to `/etc/pam.d/sshd`. When using OTPs, this symbolic link should be replaced with a private copy:

```
# cp /etc/pam.d/thinlinc /etc/pam.d/thinlinc.new
# mv /etc/pam.d/thinlinc.new /etc/pam.d/thinlinc
```

6. Configure the SSH server for RADIUS authentication by modifying its PAM configuration. The exact procedure depends on the system, but typically, this can be done by modifying `/etc/pam.d/sshd`, by inserting the line

```
auth      sufficient      /lib/security/pam_radius_auth.so use_first_pass
after the line containing pam_unix.so. Lines with references to pam_ldap in the auth section should be removed.
```

**Note:** If the system uses a "stackable" PAM configuration, then copy `system-auth` to `system-auth-radius`, and modify this file instead of `/etc/pam.d/sshd`. Then, adapt this file to use `system-auth-radius` instead of `system-auth`, in the "auth" section.

7. Restart the VSM and SSH server.
8. Login to the system with a SSH client, and verify that an OTP is required and accepted.
9. Login to the system with a ThinLinc client, and verify that an OTP is required and accepted.

#### 9.6.4. Configuration for RSA SecurID

This section describes how to deploy a OTP solution based on RSA SecurID with ThinLinc. When using this solution, the SecurID PASSCODEs are used instead of normal passwords. The PASSCODE should be entered in the ThinLinc client password input field. Please observe the following limitations:

- When SecurID requests additional information, in addition to the PASSCODE initially entered, a popup dialog will be used. This happens, for example, in Next Token or New PIN mode. After finishing the dialog, the ThinLinc client will display a "Login failed!" error message. This happens since the SBR server clears the token cache when additional information is requested. When this happens, wait until the token changes once more, and login again.
- The ThinLinc Single Sign-On mechanism will store the string entered in the clients password input field. When using SecurID, this is the PASSCODE, which cannot be used for further logins. To use the Single Sign-On mechanism, the user must be prompted for their real password. This can be done with the program **tl-sso-update-password**. To configure ThinLinc so that this program is executed during login, execute this command:

```
# ln -s /opt/thinlinc/bin/tl-sso-update-password /opt/thinlinc/etc/xstartup.d/05-tl-sso-update-pass
```

- If an external application server is used, and you want to be able to establish connections with the Single Sign-On mechanism, this application server cannot be SecurID-protected. This includes Windows Terminal Servers. When using SecurID-protected Windows domains, users and computers can be excluded from SecurID protection at the domain controller, by selecting **Control Panel**→**RSA ACE**

Agent→Domain→Advanced Domain Options . Windows Terminal Servers that are excluded in this fashion should not have the RSA Ace Agent installed.

The configuration example below assumes that you are using LDAP and RADIUS, and the Steel-Belted Radius (SBR) server. Step 8 through 11 should be repeated on all ThinLinc servers.

1. Install and configure RSA Authentication Manager (ACE server). For basic configuration tasks such as creating users and assigning tokens, we refer to the RSA documentation.
2. Create a new Agent Host for the SBR server, with type "Net OS Agent". Select which users should be able to login through ThinLinc. To allow all users, check the "Open to All Locally Known Users" checkbox.
3. Generate a configuration file for the SBR server, by selecting Agent Host->Generate Configuration File. Copy this file to `c:\windows\system32` on the machine running SBR.
4. Open the SBR Administrator. Create clients for all ThinLinc servers, using default settings. Make sure you enter a shared secret.
5. Use SBR Administrator to create a SecurID user. The user should typically be of type <ANY>.
6. Modify the SBR Authentication Policy, so that the only active method is "SecurID User". Exit SBR Administrator.
7. Enable ACE authentication caching by editing the configuration file `c:\radius\service\radius.ini` and set:  

```
[SecurID]
CachePasscodes          = yes
SecondsToCachePasscodes = 60
```

Restart SBR after changing the configuration file. For more information about ACE authentication caching, refer to the Steel-Belted Radius Tech Note RD310.
8. Install `pam_radius_auth`. Some distributions, such as SUSE, includes this module.
9. Configure `pam_radius_auth`, by creating `/etc/raddb/server` . It should contain the SBR server name, port, and a shared secret. Example:  

```
myotpserver.example.com:1812 mysecret
```
10. Configure the ThinLinc servers for RADIUS authentication by modifying its PAM configuration. The exact procedure depends on the system, but typically, this can be done by modifying `/etc/pam.d/system-auth`, by inserting the line  

```
auth          sufficient      /lib/security/pam_radius_auth.so use_first_pass
```

after the line containing `pam_unix.so`.
11. Restart the VSM and SSH server.
12. Login to the system with a SSH client, and verify that an OTP is required and accepted.
13. Login to the system with a ThinLinc client, and verify that an OTP is required and accepted.



## Chapter 10. File Access

ThinLinc supports accessing files on both Windows and Novell file servers.

### 10.1. Accessing Windows File Servers

#### 10.1.1. Introduction

This chapter describes how to setup a ThinLinc server to access Windows file servers via the SMB/CIFS protocol. CIFS is a modern version of SMB. In this document, we use the term CIFS, but the procedure described in this documentation works for SMB servers as well.

CIFS is different from NFS in that CIFS mounts are per user, not per system. For example, with NFS, it's possible to mount all network file systems when the server boots. One NFS mount can be used by all users on the system. With CIFS, each user must have their own mounts. Also, when mounting a CIFS file system, the password of the user is usually required.

ThinLinc and many other UNIX applications requires that hard links are supported in the user's home directory. There are often other POSIX file system semantic requirements as well. This means that the user's home directories cannot be a mounted CIFS filesystem. The Linux CIFS client (`smbfs`) does not support all POSIX file operations, such as hard links. The newer CIFS client (`cifsfs`) supports the CIFS UNIX extensions, but few CIFS servers support this and this feature has not been tested with ThinLinc.

ThinLinc includes two utility programs for dealing with CIFS mounts: `tl-mount-cifs` and `tl-umount-all-cifs`. These are described below.

The method described in this chapter mounts all CIFS shares below the directory `~/winshares`. The user's CIFS home directory, if any, is mounted at `~/winshares/home`.

#### 10.1.2. Requirements

##### 10.1.2.1. CIFS Server Requirements

This document assumes that you are using a Windows NT file server, version 4.0 or later (2000/XP/2003/2003 R2/2008/2008 R2). However, you should be able to use any CIFS file server.

Username and passwords must be synchronized between the file server and the ThinLinc server. Usually, this is accomplished by letting the ThinLinc servers and the CIFS file server use a common directory server. For details, please refer to Chapter 9.

##### 10.1.2.2. ThinLinc Server Requirements

Either of `smbmount/smbumount` or `mount.cifs/umount.cifs` must be installed. On Red Hat distributions, they are available in the package `samba-client`. Refer to your distribution for how to install these applications.

The programs `smbmnt/smbumount` and `mount.cifs/umount.cifs` must be `setuid root`. This is accomplished by the following commands:

```
# chmod u+s /usr/bin/smbmnt /usr/bin/smbumount
# chmod u+s /sbin/mount.cifs /sbin/umount.cifs
```

## 10.1.3. Mounting and Unmounting Shares

### 10.1.3.1. Using tl-mount-cifs

tl-mount-cifs is a small wrapper for smbmount and mount.cifs , which adds:

1. Automatically selects which file system implementation to use. `cifsfs` is used if the command `mount.cifs` is available. Otherwise, `smbfs` is tried.
2. Automatically submits password, using the ThinLinc Single Sign-On mechanism.
3. Automatically creates mount point directory, if it does not exist.
4. Can optionally fetch the service and drive letter corresponding to the users home directory specified in Active Directory.
5. Will automatically use the options specified in Hiveconf (as explained below).

The syntax for tl-mount-cifs resembles smbmount/mount.cifs :

```
tl-mount-cifs [-r] [--verbose] [-o options] service mount-point
```

```
tl-mount-cifs [-r] [--verbose] [-o options] --homedir [mount-point]
```

The `-r` option removes the mount point if the mount fails. The `--verbose` option executes both `tl-mount-cifs` and the actual mount command with debugging information. Additional mount options can be specified using the `-o options` option. Refer to the `smbmount/mount.cifs` documentation for more information. If the `--homedir` option is specified, it is not necessary to specify the service to mount. Instead, the service corresponding to the users home directory will be fetched automatically from Active Directory. This requires that the Samba `net` command is available. When `--homedir` is used, the `mount-point` argument is optional. If omitted, the service will be mounted on a directory in the users home directory corresponding to the drive letter specified in Active Directory (without the trailing colon).

The Hiveconf parameter `/utils/tl-mount-cifs/cifsmount_args` specifies default arguments for the `tl-mount-cifs` command. This Hiveconf parameter is normally found in `/opt/thinlinc/etc/conf.d/tl-mount-cifs.hconf`. The default value of this parameter is `"-o dir_mode=0700"`, which makes CIFS mounts user-private. This option is however only recognized by `mount.cifs` .

Example 1: User "john" has a home directory on the CIFS file server `\\alabama`, shared as "john\$", which should be mounted on `/home/john/winshares/home`. To do this, he runs the following command:

```
$ tl-mount-cifs -r //alabama/john$ ~/winshares/home
```

Example 2: User "john" is part of a workgroup that shares files using a share called "project" on the file server \\alabama. This share can be mounted on /home/john/winshares/project with the following command:

```
$ tl-mount-cifs //alabama/project ~/winshares/project
```

Example 3: If a home directory and home drive is specified in Active Directory, the home directory of user "john" can be executed with the command:

```
$ tl-mount-cifs --homedir
```

### 10.1.3.2. Using tl-umount-all-cifs

tl-umount-all-cifs is a utility that unmounts the current user's mounted CIFS shares (all CIFS mounts below the user's home directory). It requires no arguments. The optional argument -a will unmount all CIFS filesystems on the host.

### 10.1.3.3. Mounting Shares at Login

Often, it's convenient to automatically mount CIFS shares for all users upon login. This can be accomplished by creating a script in /opt/thinlinc/etc/xstartup.d. It can be named anything. The script should contain something like:

```
#!/bin/sh
/opt/thinlinc/bin/tl-mount-cifs //alabama/${USER}$ ~/winshares/home
```

You should also make sure that tl-umount-all-cifs runs at logout. This can be done with the following command:

```
# ln -s /opt/thinlinc/bin/tl-umount-all-cifs /opt/thinlinc/etc/xlogout.d
```

## 10.2. Accessing Novell Netware File Servers

### 10.2.1. Introduction

Novell Netware servers can serve files in several different ways. This document will focus on NCP (Netware Core Protocol) and NFS access. It is possible that information about SMB/CIFS will be added in future versions.

## 10.2.2. Using NCPFS to Access Novell File Servers

Using the `ncpfs` package and kernel module, it's possible to mount file resources from Netware servers using the Netware Core Protocol, which is the same protocol that is used by for example Windows computers accessing Netware file servers.

**Note:** There are currently two ways to access Novell File Servers via NCPFS - either via the `ncpfs` package and kernel module, or via Novell's Client for Linux. The latter is described in Section 10.2.3.

Since a NCP mount is *per user*, not per host as with NFS, every user must mount their own resources at login, most often using their password. The procedure used is much like the one used when accessing Windows file servers, described in Section 10.1.

### 10.2.2.1. Requirements

The package `ncpfs` must be installed on the ThinLinc server. The programs **`ncpmount`** and **`ncpumount`** must be setuid root. This is accomplished by executing the following command:

```
# chmod u+s,a+x /usr/bin/ncpmount /usr/bin/ncpumount
```

On SuSE, the permissions of **`ncpmount`** and **`ncpumount`** are reset each time `SuSEconfig` is run. To avoid this, add the following to `/etc/permissions.local` and run `SuSEconfig`:

```
/usr/bin/ncpmount  root.root 4755
/usr/bin/ncpumount  root.root 4755
```

The ThinLinc server must be configured to use LDAP for authentication as described in Section 9.3. The commands used for mounting resources from Netware servers, described below, use the values in `/etc/ldap.conf` to find what LDAP server to ask for needed information.

We assume that the Netware servers we will mount resources from operate in an IP environment, and that DNS names exist for all Netware servers. If the latter is not true, aliases for the Netware servers need to be added to `/etc/hosts`.

When using NCP to access the Netware file servers, the permissions are decided by the Netware server. However, the permissions that are showed on the Linux side are not mapped to the effective permission. Therefore, even though a file may look readable/writable when listing it from the Linux side, it may very well not be readable/writable when trying, because the Netware file server denies it.

### 10.2.2.2. Mounting and Unmounting shares

#### 10.2.2.2.1. Using `tl-mount-ncp`

**`tl-mount-ncp`** is a small wrapper for **`ncpmount`** which adds:

- Automatic submission of password using the ThinLinc Single Sign-On mechanism.
- Automatic creation of mountpoint directory if it does not exist.

- Automatic translation of the username into a NDS distinguished name
- Ability to automatically lookup where (on what server and path) a users home directory resides using LDAP lookups in eDirectory.

The **tl-mount-ncp** has two main syntaxes:

- **tl-mount-ncp --homedir <mountpoint>** will mount the Novell home directory of the current user (as specified in the `USER` environment variable) after looking up the location of the home directory from the `ndsHomeDirectory` attribute in eDirectory.

For example, if a user named john is logged in via ThinLinc and executes

```
tl-mount-ncp --homedir /home/john/novell/home
```

his Netware homedirectory will be mounted on `/home/john/novell/home`.

- **tl-mount-ncp <server>/<path> <mountpoint>** will mount the Netware resource at `<server>/path` on `<mountpoint>`

For example, the command

```
tl-mount-ncp ALABAMA/DATA/common /home/john/novell/common
```

will mount the directory `/DATA/common` on the Netware file server `ALABAMA` on `/home/john/novell/common`, using the current user's username and password.

If extra parameters need to be added to the `ncpmount` command line used by **tl-mount-ncp**, they can either be added on the **tl-mount-ncp** commandline using the `--ncpmount_args` parameter, or in `Hiveconf`, as the parameter `/utils/tl-mount-ncp/ncpmount_args`. The latter is the preferred way, since it provides a systemwide way to specify the parameters. By default, the parameter has a value that makes filenames with swedish characters work as expected. The `Hiveconf` parameter is found in `/opt/thinlinc/etc/conf.d/tl-mount-ncp.hconf`. You can see the `ncpmount` command line used by using the `--verbose` flag to `ncpmount` when testing functionality.

#### 10.2.2.2.2. Using *tl-umount-all-ncp*

The **tl-umount-all-ncp** command is a utility that unmounts the current user's mounted NCP shares (all NCP mounts below the user's home directory). It requires no arguments. The optional argument `-a` will unmount all NCP shares on the host.

#### 10.2.2.2.3. Mounting Shares at Login

Often, it's convenient to automatically mount NCP resources for all users upon login. This can be accomplished by creating a script in `/opt/thinlinc/etc/xstartup.d`. It can be named anything. The script should contain something like:

```
#!/bin/sh
/opt/thinlinc/bin/tl-mount-ncp --homedir ~/novell/home
/opt/thinlinc/bin/tl-mount-ncp ALABAMA/DATA/common ~/novell/common
```

You should also make sure that `tl-umount-all-ncp` runs at logout. This can be done with the following command:

```
# ln -s /opt/thinlinc/bin/tl-umount-all-ncp /opt/thinlinc/etc/xlogout.d
```

Given the example above, the user's Netware home directory will be mounted on the directory `novell/home` under the user's Linux home directory. Also, another Netware directory will be mounted at `novell/common`. Upon logout, all Netware shares will be unmounted.

### 10.2.3. Using the Native Novell Client to Access Novell File Servers

Novell has released a Linux version of their Novell Client. ThinLinc provides support scripts to help integrate the Novell Client in a ThinLinc installation by automatically adding information on username, eDirectory tree, user's context and user's password, the latter by using the ThinLinc Single Sign-On feature.

**Note:** There are currently two ways to access Novell File Servers via NCPFS - either via the `ncpfs` package and kernel module, or via Novell's Client for Linux. The former is described in Section 10.2.2.

The Native Novell Client for Linux can mount file resources from Novell file servers. This can be done manually after login by using the **map** command, or by running the user's login script. The latter is done by passing the `-r` parameter to **nwlogin**.

#### 10.2.3.1. Requirements

The Native Novell Client for Linux must be installed on the ThinLinc server. The integration scripts in ThinLinc has been tested using Novell Client v1.2 on Suse Linux Enterprise Desktop 10.

The ThinLinc server must be configured to use LDAP for authentication as described in Section 9.3. The commands used for mounting resources from Netware servers, described below, use the values in `/etc/ldap.conf` to find what LDAP server to ask for needed information.

#### 10.2.3.2. Logging in

With ThinLinc, a command named **tl-nwlogin** is shipped. This is a wrapper around the **nwlogin** command shipped as part of the Native Novell Client for Linux. The **tl-nwlogin** command takes the same commandline parameters as **nwlogin**, with the difference that if required parameters such as username, tree, context and password are missing on the commandline, **tl-nwlogin** will add them before running **nwlogin**.

If a parameter for one of several of username, tree, context or password is passed on the **tl-nwlogin** commandline, it will be passed untouched to **nwlogin**. This allows for flexibility. For example, if the user should be logged in using the same username, context and password but to a different tree, just add `-t <treename>` to the **tl-nwlogin** commandline. This will cause **tl-nwlogin** to automatically add username, context and password, but will pass the tree parameter unchanged to **nwlogin**.

### 10.2.3.3. Mounting Shares

After login, shares can be mounted by using the **map** command, part of the Novell Linux Client. They can also be mounted as part of the login procedure by adding the `-x` parameter to **tl-nwlogin**. This will cause the Novell Client to run the user's login script.

### 10.2.3.4. Logging out

After session termination, the user should be logged out to free system resources and unmount shares. For this purpose, ThinLinc ships a wrapper script named **tl-nwlogout**. This script automatically adds the `-t` parameter to tell **nwlogout** which tree to log out from.

Simply run **tl-nwlogout** by linking it into `/opt/thinlinc/etc/xlogout.d`.

If the user was logged into any additional trees, make sure you add a script for logging out of all trees in `/opt/thinlinc/etc/xlogout.d`.

## 10.2.4. Accessing Novell Network File Servers using NFS

The NFS serving capabilities in Novell Netware differ between different versions of Netware. Here's an attempt to summarize what features exist in the different variants.

**Table 10-1. NFS serving capabilities in different Novell Netware versions**

Netware version	Serving Traditional volumes	Serving NSS volumes
Netware 5.1 with Novell NFS 3.0	Support only over NFS v2. Hard links supported. File locking supported. File Access modes: NFS mode, Independent mode, Netware mode, Netware-NFS mode and NFS-Netware mode.	Supported over NFS v2 and v3. Hard links supported. File locking not supported. Available File Access modes: Independent mode.
Netware 6.0sp2+ with Native File Access for Unix	<i>Supported only over NFS v2. Hard links supported. File locking supported. File Access modes: Independent mode.</i>	Supported over NFS v2 and v3. File locking not supported. Hard links not supported. File Access modes: Independent mode.
Netware 6.5 with Native File Access for Unix	Not supported.	Supported over NFS v2 and v3. File locking supported. Hard links not supported. File Access modes: Netware mode and Independent mode.

If the filesystems exported from the Netware server is to be used as home directories for Linux users, both file locking and hardlink support is required for proper operation. Given the table above, home directories can only be exported and used from a Novell Netware server if they reside on traditional volumes on a Netware 6.0sp2 (or higher) server with Netware File Access for Unix installed.

On the other hand, using Netware mode (if your fileserver runs Netware 6.5 or later) means you don't have to assign uid and gid values to all home directories from the Linux side, since Netware will do this translation on the server side. However, since a NFS export from Netware 6.5 doesn't support hard links, the export can't be used as a home directory without special care. Also, be aware of the fact that trustees won't work, so a shared directory where several people have read/write permissions can't be used in a decent way.

### 10.2.4.1. Requirements

#### 10.2.4.1.1. Novell Netware Requirements

First you need to make sure that you have the Novell Native File Access for UNIX installed. This contains the NFS server components and NDS schema extensions. You should install the available service packs for your Netware system. Then Native File Access for UNIX is initially installed, it runs the command **schinst**, which extends the NDS schemas.

If NDS is reinstalled, if you for example installs a Netware service pack, you may have to run it manually afterwards. After **schinst** you should run **nisinst** which extends the schemas with NIS info, which is needed. You should run **nfsstop** before and **nfsstart** after these commands.

#### 10.2.4.1.2. UNIX Requirements

Of course you need some user handling, for example NIS, LDAP, NDS or similar.

### 10.2.4.2. NDS Configuration

Before you can export a Netware traditional file system you need to run the command:

```
$ add namespace nfs <filesystem>
```

Before this you won't see the file system in the list of file systems possible to export in ConsoleOne and the NFS server won't be able to export it even if you add the filesystem to `SYS:\ETC\NFSEXPRT` manually

When you have added the NFS namespace you need to export some volumes from the Netware server. In Netware 6.0 this is done using ConsoleOne. First connect to the NDS tree, and then to the NFS server. When you add the exports you can tell from which machines the root user will have root access to the files (similar to `no_root_squash` on UNIX systems). In Netware 6.5, this configuration is instead done via iManager - using ConsoleOne won't work.

If you read users and groups from NDS (by using `nss_ldap`) you should add a group, for instance `unixusers`. In the group settings you should set the `GID` in the `UNIX` -tab. Then you add your users to the group. You should also make sure that the Admin user has `UID 0` set in the `UNIX` -tab.

To get file locking you need to run the lock file daemon. You can start it with the command:

```
$ load lockd
```

You probably want to add this to your startup files to make sure that it's running from start. Also note that if you are running ZENworks for Desktops, you want to change the load order in `AUTOEXEC.NCF` so



that `NFSSTART.NCF` comes before `ZFDSTART.NCF`. Failing to do this may result in a server with 100% load and malfunctioning NFS.

### 10.2.4.3. Securing Network NFS Services

NFS can be used with several different authentication mechanisms. The most common one is called `AUTH_SYS`. This is the only mechanism supported by Netware. `AUTH_SYS` only provides a legitimate authentication if the network can be secured externally, and privileged TCP/UDP ports are used. Unfortunately, the Netware NFS server does not require that NFS requests are originating from privileged ports. This means that any user on any host (which has been granted NFS access) can impersonate any other user (including root). To work around this problem, the Netware filtering support can be used:

1. Type `FILTCFG` at the console prompt.
2. Select Configure TCP/IP Filters->Packet forwarding Filters.
3. Change status to Enabled.
4. Make sure "Deny packets in filter list" is selected.
5. Select (List of denied packets).
6. Press Insert to define a new filter.
7. Highlight the "Packet type" entry, and press Enter.
8. Press Insert to define a new packet type. It should have the following properties:
 

```
Name: nfs-unsecure
Protocol: UDP
Source ports: 1024-65535
Destination ports: 2049
Stateful filtering: Disabled
```
9. Press Escape, and then Enter to return to the Define Filter screen.
10. Press Escape and Enter to save the filter.
11. Repeat step 6 to 10 to add a TCP filter. In step 8, the Name should be "nfs-unsecuretcp", and the Protocol TCP.
12. Exit `FILTCFG`.

### 10.2.4.4. UNIX Configuration

#### 10.2.4.4.1. Assigning permissions to NFS-exported filesystems

When exporting filesystems from a Novell Netware fileserver using NFS in the Independent mode (see Table 10-1 for the available file access mapping modes on different Netware versions), no mapping is done from the Netware permissions to the Unix permissions. Therefore, permissions on the exported filesystems must be set from a Linux client by running `chmod` as root. This can be done either manually or automatically using `tl-nds-posixuser`, described in Section 9.3.4.

#### 10.2.4.4.2. Mounting the homedirectories on the ThinLinc server

Depending on how the NFS-exports of filesystems on the Netware servers are organized, it's more or less easy to mount the homedirectories on the ThinLinc server(s).

##### 10.2.4.4.2.1. All homedirectories on one export

If all homedirectories are found under one export, just add a line to `/etc/fstab` that mounts that path as `/home`. This situation is however rare. Remember to put `nfsvers=2` if the server only exports using NFS v2. Another option is let autofs mount the home directories using a wildcard entry in the automounter map.

##### 10.2.4.4.2.2. Homedirectories spread among several servers and/or mountpoints

If the homedirectories are spread among several different servers and/or on several different exports, it's less easy to mount the correct path. For this purpose, ThinLinc provides the program **tl-nds-mountpath** that searches LDAP and deducts the correct mountpath from the `ndsHomeDirectory` value.

The **tl-nds-mountpath** is added on a line in `/etc/auto.master`. An example line looks like this:

```
/home          program:/opt/thinlinc/libexec/tl-nds-mountpath nfsvers=2,hard,intr
```

This will cause **tl-nds-mountpath** to search LDAP for the correct mountpath for a specific user. The program reads its parameters from `/etc/ldap.conf` or `/etc/pam_ldap.conf`, that is, it's using the same DN and password as `pam_ldap` and `nss_ldap`. Make sure the DN have the permissions to read the `ndsHomeDirectory` attribute or **tl-nds-mountpath** will not work as expected. Use the ThinLinc Novell Configurator as documented in Section 9.3.1 to create and configure the special DN used to search LDAP for the needed information.

**Note:** In some Linux distributions, the `/etc/auto.master` syntax above will not work. Instead, the line should look like this:

```
/home          program /opt/thinlinc/libexec/tl-nds-mountpath nfsvers=2,hard,intr
```

SUSE Linux Enterprise Server is one example where this syntax is required.

Since many eDirectory installations has one or several entries where the `ndsHomeDirectory` has an invalid value, **tl-nds-mountpath** normal behaviour is to check the result of the LDAP query to make sure the key is also in the result. For example, when asked for the mountpath for the home directory of the user *alice*, it checks that the result contains the string *alice*, in a case-insensitive string search. This means that if the environment is setup so that the username is not in the home directory path, **tl-nds-mountpath** will not work. If this is not the desired behaviour, it can be turned off by setting `/utils/tl-nds-mountpath/key_in_result_verify` to 0.

The filter used by **tl-nds-mountpath** to find the user object and hence the `ndsHomeDirectory` attribute can be configured in the parameter `/utils/tl-nds-mountpath/ldapfilter`. The default filter removes alias objects from the result.

The reason for this behaviour is that when **tl-nds-mountpath** is used in combination with **tl-nds-posixuser**, assigning uid/gid values, a whole directory with many home directories can be assigned incorrect uid/gid values, if the `ndsHomeDirectory` attribute of one user has a bad value.

Netware file systems are case insensitive when using NCP and CIFS, but case sensitive when using NFS. Often, the path specified in NDS has a different case than the actual directory. If you try to mount the path as specified in NDS, you will get an error message. The best way of solving this is to enter the correct home directory path in NDS. However, correcting this for many users can be time consuming. Therefore, **tl-nds-mountpath** contains support for testing different combinations of uppercase and lowercase characters. When it finds a combination that works, the result is returned to the automounter. Since the number of possible combinations is typically very large, not all combinations can be tested. The combinations to test is specified by the parameter

`/utils/tl-nds-mountpath/directory_cases`. It is a list of strings. Each string can contain the letters l, u or n, for lowercase, uppercase and NDS, respectively. These strings specify the different case combinations for the beginning of the name of directories. The default value is "n l u ul". This means that **tl-nds-mountpath** tries all combinations of directories that are named like:

- Just like it is specified in NDS
- With all letters in lowercase
- With all letters in uppercase
- With the first letter in uppercase, and the rest in lowercase

Other variants can be added as well, but please note that if you have a deep directory structure and many case strings, the number of different combinations will be very large, which means that the execution of **tl-nds-mountpath** might take a long time. The number of different combinations  $k$  is  $s^{d-1}$ , where  $s$  is the number of case strings, and  $d$  is the number of directories in the home directory path. The average search time  $t$  is  $k * 0.005 / 2$  seconds (assuming one mount requests takes 5 ms). The table below lists the search time for some sample values of  $s$  and  $d$ .

**Table 10-2. Combinations and average search time for tl-nds-mountpath**

Directories in home directory path:	1	2	3	4
1 case string	1 (5 ms)	1 (5 ms)	1 (5 ms)	1 (5 ms)
2 case strings	1 (5 ms)	2 (5 ms)	4 (10 ms)	8 (20 ms)
3 case strings	1 (5 ms)	3 (7.5 ms)	9 (22.5 ms)	27 (67.5 ms)
4 case strings	1 (5 ms)	4 (10 ms)	16 (40 ms)	64 (160 ms)
5 case strings	1 (5 ms)	5 (12.5 ms)	25 (62.5 ms)	125 (312.5 ms)
6 case strings	1 (5 ms)	6 (15 ms)	36 (90 ms)	216 (540 ms)

#### 10.2.4.4.3. Special considerations when using directories mounted from Netware 6.5 as home

### directories

Since exports from Netware 6.5 don't support hard links, they cannot be used as home directories in Linux without special care. Both the X Window System, Gnome and KDE use hard links when locking files. One workaround for this is to perform the following steps.

- Create a directory either on the local filesystem or, in the case of a ThinLinc cluster, on the machine running the VSM server. Regardless of which, we will call it `/localhome` in this example. Make `/localhome` sticky by executing **chmod o+ws /localhome**. This will enable all users to create files and directories in `/localhome`, but no user will be able to erase another user's files/directories.
- Each time the user logs in, check if the directory `/localhome/<username>` exists. If not, create it using **mkdir**.
- Set the following environment variables for each user:

```
ICEAUTHORITY=/localhome/${USER}/.ICEAuthority
GCONF_LOCAL_LOCKS=1
```

**Note:** It is only necessary to set `GCONF_LOCAL_LOCKS` for GConf 2.3.0 or older. In newer versions, you must instead make sure that the variable `GCONF_GLOBAL_LOCKS` is not set to 1.

Another problem is that in Netware, most home directories are not owned by the user it is associated with, but instead some superuser owns it (since the superuser created it). Try using **FLAG.EXE** from Windows to correct this.

## 10.3. Restricting write access to users home directory

### 10.3.1. Introduction

When accessing directories from CIFS and NCP servers, these are mounted in subdirectories of the users UNIX home directory. It is not possible to place the UNIX home directory on a CIFS or NCP server, since these typically do not support the necessary POSIX file system semantics (such as hard links). In a typical setup, applications such as Mozilla use the UNIX home directory for settings (`~/.mozilla`), while the user saves documents in `~/MyDocuments`. In this case, it might be desirable to restrict access to the UNIX home directory: Forbid saving arbitrary files to it. This can be solved by using a feature of ThinLinc called `homecreatefilter`.

### 10.3.2. Activation

To activate `homecreatefilter`, create a symbolic link in the `xstartup.d` directory:

```
# ln -s /opt/thinlinc/libexec/tl-homecreatefilter.sh /opt/thinlinc/etc/xstartup.d/06-tl-homecreatefilter.sh
```

### 10.3.3. Configuration

The configuration file `/opt/thinlinc/etc/homecreatefilter.conf` controls which files and directories are allowed. By default, all files starting with a dot are allowed, as well as the files necessary for KDE to start.

The configuration file is line based. A line not starting with a colon specifies a file object pattern that should be allowed. A line starting with a colon specifies a command line pattern. Processes matching this pattern will also be allowed write access, even if no file object pattern allows access.

### 10.3.4. Security Considerations and Limitations

The homecreatefilter feature is based on the LD\_PRELOAD mechanism, which means it does not support statically linked applications. Since environment variables can be modified by the user, the user can disable the filter at will. homecreatefilter should not be regarded as a security mechanism, but rather a mechanism that prevents the user from saving documents to the UNIX home directory by mistake.

In addition to the home directory, homecreatefilter restricts write access to the `~/Desktop` directory.



# Chapter 11. Connecting to Windows Terminal Servers

## 11.1. Introduction

This chapter describes how to connect to Windows Terminal Servers via RDP. This makes it possible to provide ThinLinc users with Windows desktops, but also to publish individual Windows applications to a ThinLinc desktop, running on Linux or Solaris.

It is also possible to connect to Citrix servers by installing the Citrix ICA Client on the ThinLinc servers. In this case, the ThinLinc commands **tl-wfica** and **tl-wfcmgr** can be used to provide Single Sign-On. For more information about **tl-wfica** and **tl-wfcmgr**, see Chapter 13.

## 11.2. Single Sign-On

### 11.2.1. Information

ThinLinc provides Single Sign-On functionality into the Windows Terminal Server using either password or smart card authentication. It is required that your ThinLinc servers are integrated with your Windows infrastructure so that user authentication shares the same source on both Windows and ThinLinc.

If requirements mentioned above are met, Single Sign-On works out of the box with one exception regarding smart card and CredSSP which is documented in the following section.

### 11.2.2. Smart card

If you want to use smart card Single Sign-On using Windows 2003 you need to install wstools package with our ThinLinc GINA. See Section 3.7 for more information.

Windows 2008 (RDP v6) and later does not require the mentioned GINA for using smart card SSO authentication.

If your Windows Terminal Server is configured to explicitly only allow CredSSP authentication level, ThinLinc needs to know a provide name for your smart card Crypto Service Provider (CSP). The provider name is configured per application server group and is added to `rdesktop_args` configuration value like the example below. See Section 14.2.4 for more information.

**`rdesktop_args=-o sc-csp-name="CSP Provider Name"`**,

To obtain the provider name of your Crypto Service Provider (CSP) make sure that your smart card driver are installed on your Windows Terminal Server. Open regedit and find the following registry key, `HKLM\SOFTWARE\Microsoft\Cryptography\Defaults\Provider`. In this container you will find a list of CSP providers registered with the system, find the matching provider for your smartcard and use the key name as the CSP Provider Name.

## 11.3. Connection Modes

This section describes the different connection modes, and lists their limitations.

### 11.3.1. Running a Windows Desktop in a Window

Sometimes it's useful to run a Windows desktop in a window. This mode resembles many virtualization solutions. The distinction between the Windows and UNIX environment is obvious. Connections of this type can be made by using the **tl-run-rdesktop** command.

### 11.3.2. Running a Windows Desktop in Fullscreen

In this mode, the Windows desktop fully replaces the UNIX desktop. The UNIX environment is completely hidden. Connections of this type can be made by using the **tl-run-windesk** command.

### 11.3.3. Running a WTS application in Standard Mode

The Standard Mode provides a limited form of application publishing. No extra software on the Windows Terminal Servers is required. Typically, connections of this type are created with a command such as: **tl-run-winapp -D -T Excel excel.exe**. When using this mode, please note:

- The RDP connection window cannot be resized: It will remain the same size throughout the entire RDP session. The default size is to occupy the "work area": The desktop area not covered by desktop panels. Another size can be selected by using the **-g** argument.
- It is possible to minimize the application. When using a size smaller than the work area, it is also possible to move the application by using the applications title bar. Both these features requires that a standard graphical theme is used on the Windows Terminal Servers: The button size should be 18 pixels. If not, the button size can be specified by using the **-S** option. For more information, consult the **rdesktop** documentation.
- The application must use a single, maximized window. The Windows calculator does not work correctly, since it cannot be maximized. Mozilla Firefox does not work correctly when using multiple windows.

For more information about **tl-run-winapp**, see the **tl-run-winapp** details in Chapter 13.

### 11.3.4. Running a WTS application in SeamlessRDP Mode

This mode makes it possible to publish applications "seamlessly", and allows for full integration with the desktop. The remote applications can be moved, resized and restacked. Support for non-maximized and multi-window applications is provided. When using the SeamlessRDP mode, the "WTS Tools" package must be installed on all Windows Terminal Servers. The installation is described in Section 3.7.

Typically, connections of this type are created with a command such as: **tl-run-winapp-seamless c:\program\mozilla.org\mozilla\mozilla.exe http://www.cendio.com**. When using this mode, please note:

- A modern window manager is highly recommended.
- No support for the System Tray is provided.
- The command line application (cmd.exe) is not supported.



- You cannot run Internet Explorer in Protective Mode.
- It is not possible to launch Explorer in file manager mode. As an alternative, Internet Explorer can be used.

For more information about **tl-run-winapp-seamless**, see the tl-run-winapp details in Chapter 13.

**Note:**

- Windows 7 does not support publishing applications in SeamlessRDP mode.
- Windows Server 2012 and Window Server 2012 R2 does not support SeamlessRDP mode.



## Chapter 12. Accessing Client Resources from the Terminal Server

In this chapter we will describe how to access client resources, such as local drives and serial ports, from the Terminal Server.

### 12.1. Accessing the Clients Local Drives

#### 12.1.1. Introduction

Using ThinLinc, it is possible to access the clients' drives and filesystems from the terminal servers. With thin terminals, one might want to access a local CD-ROM drive. When running the client on a workstation, applications on the terminal server can access all filesystems mounted at the workstation, just like local applications can.

**Note:** Many Digital Cameras can be accessed as a USB storage device, and can be exported as a local drive.

#### 12.1.2. Mounting and Unmounting Local Drives

The exported local drives can be mounted with the command `tl-mount-localdrives`. The drives will be mounted below `$TLSESSIONDATA/drives`. A symbolic link called "thindrives" will be created in the user's home directory, pointing to this directory. The syntax for `tl-mount-localdrives` is:

**tl-mount-localdrives** [-h] [-v]

The `-v` option causes the tool to be executed in verbose mode, while `-h` shows the syntax.

The Hiveconf parameter `/utils/tl-mount-localdrives/mount_args` specifies the mount arguments. This Hiveconf parameter is normally found in `/opt/thinlinc/etc/conf.d/tl-mount-localdrives.hconf`. The options `mountport`, `port`, `mountvers`, `nfsvers`, `nolock`, and `tcp` will always be used.

Mounted local drives can be unmounted with the command `tl-umount-localdrives`. If some applications are using a mount at this time, they can continue to access the mount, even though the mount has been removed from the filesystem hierarchy (so called "lazy" umount). The syntax for `tl-umount-localdrives` is:

**tl-umount-localdrives** [-a] [-s] [-l]

If `-a` is specified, then all mounted local drives, for all users on this machine, will be unmounted. If `-s` is specified, then all mounted local drives, for all sessions belonging to the current user, will be unmounted. If `-l` is specified, the `thindrives` link will not be updated.

**Note:** When using multiple sessions per user, the `thindrives` link will point to the newest session that executed **tl-mount-localdrives**. **tl-umount-localdrives** will restore the link to the newest session which is not newer than the current session and which has mounted local drives.

### 12.1.3. Accessing local drives from Windows Terminal Servers

When using Windows Server 2003 or newer, access to the Local Drives is possible via RDP. In this case, no configuration is required; sessions started with `tl-run-rdesktop` (and associated commands) will automatically have the local drives redirected.

**Note:** Due to what seems to be a limitation in the RDP protocol, local drives with names longer than seven characters will be displayed using only the first seven characters on the Windows Terminal Server.

Local Drives can also be accessed from Citrix ICA servers. In this case, the Citrix ICA client must be configured to export the desired "thindrives" directories.

### 12.1.4. Mounting Drives at Login

Often, it's convenient to automatically mount all local drives for a user when the session starts. This is done by default via a symbolic link in `/opt/thinlinc/etc/xstartup.d`, pointing at `/opt/thinlinc/bin/tl-mount-localdrives`. This link is created for you during installation, as well as its counterpart in `/opt/thinlinc/etc/xlogout.d` which points to `/opt/thinlinc/bin/tl-umount-localdrives`.

### 12.1.5. Limitations and additional information

- A mounted local drive, for example `/var/opt/thinlinc/sessions/joe/47/drives/cdrom`, is only usable during the lifetime of the ThinLinc session. If the user ends the session without unmounting and then starts a new session, the mount will not be usable even if the session number happens to be same. In this case, any attempts to access the mount will give the error message "Stale NFS file handle". To be able to use the local drive, the user needs to run `tl-mount-localdrives`.
- The mounted local drive does not fully support POSIX semantics. The usual limitations of NFSv3 applies. Additionally, if the file is moved to another directory while a process has the file open, the process will get a "Stale NFS file handle" error on any subsequent file operation for that file.
- Local files are uniquely identified by their inode number. Some file system implementations, such as the Linux kernel FAT implementation, do not provide persistent inode numbers. Inode numbers will change on each remount, which usually results in "Stale NFS file handle" errors.
- To prevent users from mounting local drives at the terminal server, execute this command:  

```
# chmod u-s /opt/thinlinc/libexec/tl-mount-personal
```

## 12.2. Using Serial Port redirection

### 12.2.1. Introduction

Using ThinLinc, it is possible to access the serial ports of the client from the terminal servers. This means that you can utilize peripheral devices which connect through a serial port, such as digital cameras, PDAs and modems. Up to two serial ports are supported at a time.

### 12.2.2. Requirements

- The application which communicates with the serial port must be dynamically linked. Statically linked applications are not supported.

### 12.2.3. Enabling Serial Port Redirection

Serial port redirection is activated (for the current user session) by sourcing the file `tl-serial-redir.sh`. It can be done manually with this command:

```
$ source /opt/thinlinc/libexec/tl-serial-redir.sh
```

It is necessary to source this file, because it sets the environment variables `CYCLADE_DEVICES` and `LD_PRELOAD`. Thus, all applications needing serial port access should be started as a subprocess to this shell. The easiest way to accomplish this is to source `tl-serial-redir.sh` from the session startup scripts. To automatically activate serial port redirection at login for all users, execute this command:

```
# ln -s /opt/thinlinc/libexec/tl-serial-redir.sh /opt/thinlinc/etc/xstartup.d/42-tl-serial-redir.sh
```

### 12.2.4. Accessing the redirected port from applications

When using redirected serial ports, applications should be configured to use a special, personal device-file, instead of a port such as `/dev/ttyS0`. The two device files are called `$TLSESSIONDATA/dev/ttyS0` and `$TLSESSIONDATA/dev/ttyS1`.

**Best Practice:** Since the session number varies, it's often convenient to use the symbolic link `/var/opt/thinlinc/sessions/$USER/last`, which points to the last started session directory. For example, the first serial port can be accessed as `/var/opt/thinlinc/sessions/$USER/last/dev/ttyS0`.

### 12.2.5. Limitations and additional information

- When reconnecting to an existing session, it might take up to 10 seconds before the serial ports are available.
- A maximum of two serial ports per session can be redirected.
- Currently, the redirected serial port is not accessible from Windows Terminal Servers. RDP serial port redirection will be supported in future ThinLinc releases.
- The redirection is handled by processes called `cyclades-ser-cli`. It writes debugging information to `$TLSESSIONDATA/ttyS0.log` and `$TLSESSIONDATA/ttyS1.log`. These processes will automatically terminate when the session terminates.
- Applications that uses the `ioctl TIOCMGET` are not supported yet.

## 12.3. Using Sound Device Redirection

### 12.3.1. Introduction

With ThinLinc, it is possible to access the client's sound device from the terminal servers. This means that you can run sound applications on the terminal servers and listen to the sound through the client's sound device and speakers. Input devices such as microphones can also be used.

### 12.3.2. Requirements

- EsoundD client libraries to support applications with native EsoundD support.
- esddsp to support OSS applications via EsoundD.
- PulseAudio client libraries to support applications with native PulseAudio support and the ALSA plug-in. ThinLinc supports version 0.9 of PulseAudio.
- padsp to support OSS applications via PulseAudio.
- alsa-plugins, version 1.0.12 or later, to support ALSA applications via PulseAudio.

### 12.3.3. Using sound redirection with UNIX applications

ThinLinc can support sound redirection for almost all applications, provided that the correct libraries and utilities are installed on the ThinLinc server.

#### 12.3.3.1. EsoundD applications

All applications that can communicate using the EsoundD protocol will work automatically in ThinLinc. Most modern multimedia applications support EsoundD, meaning that the only required work is to make sure they use EsoundD by default. Unfortunately, EsoundD does not work well for users running multiple ThinLinc sessions.

### 12.3.3.2. PulseAudio applications

All applications that can communicate using the PulseAudio protocol will also work automatically in ThinLinc. Most current distributions are configured to use PulseAudio by default, but older ones might require some configuration to work properly.

### 12.3.3.3. OSS applications

Most applications that use the Open Sound System (OSS) can be made to work with ThinLinc through the esddsp or padsp application.

padsp redirects OSS applications to the PulseAudio protocol. The following command line should be used:

```
padsp <application>
```

See the padsp manual page for more information.

esddsp redirects OSS applications to the EsoundD protocol. The command line is similar to padsp but requires an extra -m to enable mixer emulation:

```
esddsp -m <application>
```

See the esddsp manual page for more information.

The application which communicates with the sound device must be dynamically linked to glibc. It is not possible to intercept the accesses to OSS in a statically linked application. Most applications that you find on a Linux system will satisfy this requirement, but a test with ldd can also be done:

```
$ ldd /usr/bin/someapp
      not a dynamic executable
```

When using esddsp or padsp on 64-bit platforms, make sure that you have both 32- and 64-bit versions of the necessary libraries (`libesddsp.so.0` and `libesd.so.0` for esddsp, `libpulsedsp.so` and `libpulse.so.0` for padsp). Usually, these are found in `/usr/lib` and `/usr/lib64`. Also, the esddsp and padsp scripts must not contain absolute references to these libraries. Instead, the system should automatically select the correct library, depending on if you are executing a 32- or 64-bit application. In this case it's necessary to have both library directories included in `/etc/ld.so.conf`.

esddsp has a bug which require you to create some files in `/tmp` in order for it to work. Execute the following as root:

```
mkdir /tmp/.esd
touch /tmp/.esd/socket
```

Many distributions clean out `/tmp` periodically so this procedure should be added to that script or repeated as necessary.

Although it is rare, some applications manage to misuse the OSS API in a way that works with local sound cards but not esddsp or padsp. If you encounter problems, try updating the application to the latest version as it might contain fixes for such bugs.

#### 12.3.3.4. ALSA applications

All applications that use the Advanced Linux Sound Architecture (ALSA) will also work well with ThinLinc provided the correct plug-ins are installed and configured. The plug-ins can be found in the `alsa-plugins` package (called `libasound2-plugins` on Debian-based distributions). The PulseAudio client libraries are also needed to build and use the plug-ins.

To redirect ALSA applications to use the plug-ins, the ALSA configuration must be modified. This can be done on a global level in `/etc/asound.conf` or per user in `~/.asoundrc`. Add the following to the file (creating it if necessary):

```
pcm.!default {
    type pulse
}
ctl.!default {
    type pulse
}
```

Unfortunately, there are some applications that use the ALSA API in an incorrect way. When using local hardware this usually doesn't matter, but when advanced ALSA features, like `dmix` or this plug-in, are used, then problems start to appear. If an application misbehaves, the first step should be to upgrade it to the latest version. With some luck, the API is used more correctly in a later version.

#### 12.3.3.5. Choosing sound system

Many applications support several sound systems and it can be difficult to know which one to use. The general principle is to choose a PulseAudio solution over a EsoundD one at all times.

Applications should be configured in the following manner, listed from the best solution to the worst:

1. Native PulseAudio application.
2. ALSA application using the PulseAudio plug-in.
3. OSS application using `padsp` .
4. Native EsoundD application.
5. OSS application using `esddsp` .

#### 12.3.4. Using sound redirection with Windows Terminal Servers

It is possible to use sound redirection with applications running on Windows Terminal Servers, via RDP. This is controlled by the parameter `/appservergroups/rdp/<group>/sound` . For more information, please refer to Section 14.2.4.

To use input devices with applications running on Windows Terminal Servers, it is necessary to use the ThinLinc WTS sound driver. See Section 3.7 for more information.

Sound redirection can also be used in conjunction with the Citrix ICA client, which is an OSS application. For more information, see the section about OSS applications .



### **12.3.5. Limitations and additional information**

- Transferring sound over the network requires a lot of bandwidth, so it is only suitable for high-speed networks, such as LANs.

## **12.4. Using Smart Card Redirection**

### **12.4.1. Introduction**

Using ThinLinc, it is possible to access the locally connected smart cards and smart card readers from the terminal servers. This means that you can use smart cards for encrypting your email, signing documents and authenticating against remote systems.

### **12.4.2. Requirements**

- The application which communicates with the smart card must be using the PC/SC API and be dynamically linked to pcsc-lite.

### **12.4.3. Enabling Smart Card Redirection**

Smart card redirection is always activated on the server so there is no administration required to enable it.

### **12.4.4. Limitations and additional information**

- When a client disconnects, all smart cards and smart card readers will disappear for the applications. Some applications do not handle hot-plug and must therefore be restarted when this happens.



## Chapter 13. Commands on the ThinLinc Server

In this chapter, we will describe the commands shipped as part of the ThinLinc server that are meant for the common user.

### Commands in `/opt/thinlinc/bin`

**tl-best-winservice** *server* [*server ...*]

The **tl-best-winservice** command asks the Windows Terminal Servers listed on its commandline for their respective load status. It then prints the name of the least loaded terminal server on standard out, and exits. If the file `.thinlinc/last-winservice` exists in the user's home directory, the server listed there will be checked for sessions owned by the user. If such a session exists, **tl-best-winservice** will print the name of that terminal server regardless of its load, since the user should get his/her old session when logging in again. The information that this script prints out is used by **tl-run-rdesktop** when it chooses which Windows Terminal Server to connect to.

**tl-session-param** [*options*] *parameter*

The **tl-session-param** command is used to access the session information managed by the VSM server. This includes information sent by the client, such as if the client has exported any local drives, or what language is set on the client side. This command is used by for example **tl-set-clientlang.sh**, documented later in this chapter.

**tl-config** *options*

The **tl-config** command is used to access configuration parameters used by the ThinLinc system. It is also used to set parameters from scripts, and can be used instead of an editor when some parameter needs to be changed. **tl-config** uses **hivetool**, part of the Hiveconf system. See Chapter 15 for more information about Hiveconf.

**tl-desktop-restore**

When a user's Gnome or KDE desktop needs to be reset to default, the command **tl-desktop-restore** can be run. This will move the settings directories for KDE and Gnome to a backup directory named `.old-thinlinc-desktop` in the user's home directory, which will make both Gnome and KDE revert to the default settings.

### **tl-limit-printers**

This command is run by VSM Server at session startup and reconnect if the Printer Access Control feature of ThinLinc is activated. See Section 5.5 for details.

### **tl-mount-cifs**

This command is used to mount CIFS/SMB network file systems at login-time. See Section 10.1 for documentation on this subject.

### **tl-memberof-group** *groupname*

This command can be used to determine if the current user is a member of the specified group. It returns true (0) if the user is a member, false (1) if the user is not a member and false (2) if the specified group does not exist.

### **tl-nds-memberof-container** *LDAP container*

This command can be used to determine if a user is member of a specific LDAP/eDirectory container, in eDirectory known as ou. It returns true (0) if the LDAP object corresponding to the current user resides under the container specified as parameter and false(1) if not.

This command connects to the LDAP/eDirectory server using the parameters specified in `/etc/ldap.conf`.

### **tl-passwd**

This command is used to let the user change their password, both in the underlying authentication mechanism and in the ThinLinc Single Sign-On mechanism.

In order for this to work, any user must be able to read the file `/etc/pam.d/sshd` (or, more correct, the file that the symbolic link `/etc/pam.d/thinlinc` points at).

Also, in the case where the underlying authentication mechanism is LDAP or eDirectory, make sure that the parameter `pam_password` in `/etc/ldap.conf` is set to a value that is appropriate for your environment. If you're authenticating against eDirectory servers, it must be set to `nds`. See the comments in `ldap.conf` for more information.

**tl-run-rdesktop** [*options*]

The **tl-run-rdesktop** program is a wrapper around the **rdesktop** program. It extends the functionality of **rdesktop** by connecting to one of the Windows Terminal Servers specified by the system administrator in `/appservergroups/rdp/<group>/servers`. If the user has a pre-existing session on one of the servers in the list, the session is reconnected. If not, the server with the least load is selected. The command creates the connection with the correct domain and keyboard layout. For more information, see Section 14.2.4. Multiple groups of Windows Terminal Servers can be specified. This makes it possible to direct different groups of users to different Terminal Servers in an easy way. In the parameters specified above, exchange `<group>` for the group you have specified in Hiveconf, and tell **tl-run-rdesktop** which group to use by using the `-G` commandline option. If no group is selected, the group named `default` is used.

**tl-run-unixapp** [*arguments*]

This command uses single sign-on to login to a UNIX server defined in `/appservergroups/x11/<group>/servers`, executing either a shell or the commands specified. It also sets up X11 over SSH if defined in `/appservergroups/x11/<group>/use_ssh_encryption`. Just as with **tl-run-rdesktop**, **tl-run-unixapp** supports application server groups, which means multiple groups of Unix servers can be specified. The application server group to be used is chosen by using the `-G` commandline option to **tl-run-unixapp**. If no group is selected, the group named `default` is used.

**Note:** This command requires that the OpenSSH client is installed on the server where it is run.

**Best Practice:** If the SSH host key of the server **tl-run-unixapp** is configured to connect to is not known, a window will be shown where the user is asked if the host key should be trusted. If this question is confusing your users, add the host keys of the servers in `/etc/ssh/ssh_known_hosts`. That will make SSH recognize the host, removing the question for the users. It will also increase security, since the host key is then checked by personell that have the ability to actually verify the key.

**tl-run-winapp** [-D] [-T *title*] [*arguments*] *windows-app* [*application arguments*]

This is a wrapper around **tl-run-rdesktop** that executes a single command on the Windows Terminal Server with most system resources left.

**tl-run-winapp** takes many different arguments, but the most common ones are `-D`, which hides the window manager decorations. `-T title` sets the title of the window to the string in question.

**Example:** `tl-run-winapp -D -T Excel excel.exe`

**tl-run-winapp-seamless** [*arguments*] *windows-app* [*application arguments*]

This command resembles **tl-run-winapp** except that the application is executed in SeamlessRDP mode. This allows for full integration with the desktop. For more information, see Section 11.3.4.

**tl-run-winapp-seamless** takes the same arguments as **rdesktop**, except for the `-G` parameter.

**Example:** `tl-run-winapp-seamless c:\program\mozilla.org\mozilla\mozilla.exe  
http://www.cendio.com`

#### **tl-run-windesk**

This is a wrapper for **tl-run-rdesktop** that starts a full screen session against the best Windows Terminal Server available.

#### **tl-run-xstartup.d**

This command is run by the session startup file (`~/thinlinc/xstartup`) in its default form to execute all start scripts in the directory `/opt/thinlinc/etc/xstartup.d/`. Files with the suffix `.sh` will be sourced. All other files will be executed.

#### **tl-select-profile**

This command is run by the session setup file (`/opt/thinlinc/etc/xstartup.default` or `~/thinlinc/xstartup`) and provides a menu where the user can choose what kind of session to run. See Section 14.4 for more information.

#### **tl-set-clientlang.sh**

By creating a symlink from `/opt/thinlinc/etc/xstartup.d` to this command, the user's LANG environment will be set to the language environment reported by the client.

**tl-shadow-notify**

This command starts the **tl-shadow-notify** command for the lifetime of the session. This will enable notifications when the session is shadowed.

**tl-single-app** *command* [*arguments*]

The **tl-single-app** command can be used to execute a single application in a ThinLinc session. A window manager with a suitable configuration is automatically started. All top level windows are automatically maximized. Window titles are displayed in the title bar of the ThinLinc Client, not in the ThinLinc session. The client close button will disconnect the session as usual. Inner close buttons closes application windows. The **tl-single-app** command can be specified as a client supplied start program (see Section 14.4.4), or used with the ThinLinc profile selector (see Section 14.4.5).

**Switching Between Windows:** If the application opens multiple top level windows, you can switch between them by clicking on the application icon in the top left corner.

**tl-sso-update-password**

This command requests a password from the user, to be used with the Single Sign-On mechanism of ThinLinc. It is useful when the password is not already available, for example, when using One Time Passwords. See Section 9.6.4 for more information.

**tl-support** [-p *listen-port*] [-u *user*] [*host*]

The **tl-support** command can be used to enable a support technician to login to your ThinLinc server, even though the server is behind a firewall that doesn't allow connections to the ssh port. This is accomplished by opening a ssh connection from the server to an external server on the internet, at the same time setting up a tunnel from the remote host to the local host's ssh port. The default server to connect to is support.thinlinc.com with the default username "support". This command should only be used after contacting your ThinLinc support technician.

**tl-umount-all-cifs**

This command is used to unmount CIFS/SMB network file systems at logout-time. See Section 10.1 for documentation on this subject.

**tl-wfica** [*options*] [*connection file*]

This command is a front-end for the Citrix ICA client (**wfica**), which provides Single Sign-On for ICA sessions in a secure fashion. This command does not modify any ICA Client configuration files. The ThinLinc Single Sign-On password is only used for connection which lacks a manually configured password. The arguments are the same as for **wfica**. For more information, consult the Citrix ICA Client documentation.

**tl-wfcmgr** [*options*] [*connection file*]

This command is a front-end for the Citrix ICA client GUI (**wfcmgr**), which provides Single Sign-On for ICA sessions in a secure fashion. Single Sign-On is provided both for the "Connection" and "PNAgent" view, through modifications to `~/.ICAClient/appsrv.ini` and `~/.ICAClient/reg.ini`. Only connections which lacks a manually configured password are modified. Single Sign-On for the "PNAgent" view requires that the Citrix server allows the user to save passwords.

### Warning

When using **tl-wfcmgr**, the ThinLinc Single Sign-On password is permanently stored in `~/.ICAClient/appsrv.ini` and `~/.ICAClient/reg.ini`.

The arguments are the same as for **wfcmgr**. For more information, consult the Citrix ICA Client documentation.

**tl-disconnect**

This command is used to disconnect from the current session. This can be used to provide an alternative to the F8 key, such as a disconnect button on the Gnome panel.

**tl-sso-password** [--check] [--remove]

This command can be used to hook up the Single Sign-on mechanism of ThinLinc with new applications. It can be used to test for the presence of a valid password and to feed that password out on standard output to another application.

To check for the existance of a valid password, invoke the command as **tl-sso-password --check**. A return code of zero indicates a valid password.

If the **--remove** option is specified, the password will be removed, after the retrieval or check.

There are two basic models to connect **tl-sso-password** to an application. The first is to use shell pipes:



```
# tl-sso-password | /usr/bin/application --read-password-on-stdin
```

The second is to have the application invoke **tl-sso-password** as needed:

```
# /usr/bin/application --password-prog tl-sso-password
```

**tl-sso-token-passphrase** [--check] [--remove]

This command is identical to **tl-sso-password**, except that it uses the smart card token passphrase (PIN) instead of the user's password. For usage, see the **tl-sso-password** section above.

**tl-env** [-d] [-n *nr*] [*command* [*arg...*]]

**tl-env** [-s] [-n *nr*]

This command can be used to save and restore the ThinLinc session environment variables. It operates on the file `xstartup.env` in the session directory. During session startup, **tl-env** is called with the `-s` option after everything in `xstartup.d` have been executed. Later, **tl-env** can be used to execute a command in this environment, even outside the ThinLinc session. During restore, the `DISPLAY` environment variable can be excluded by specifying `-d`. By default, this command operates on the "last" session number for the invoking user. An alternative session number can be specified with the `-n` option.

## Commands in `/opt/thinlinc/sbin`

**tl-notify** [-u *username*] *message*

This command sends a user-visible message to ThinLinc sessions on the server. The default is to send the message to all sessions, but the `-u` option can be used to send the message to a single recipient instead.

To send messages to all users in a ThinLinc cluster, you can use this command in combination with the **tl-ssh-all** command described in this section.

**tl-rsync-all**

This command is used to synchronize files and directories in a ThinLinc cluster. It runs the `rsync` command over SSH against all terminal servers in the cluster. When using this command, it's convenient if password-less SSH login between the servers in the clusters has been setup.

See also **tl-ssh-all** below for some tips regarding password-less running of `ssh`.

## tl-ssh-all

This command is used to perform shell commands on all slaves in a ThinLinc cluster. It works by running the ssh command against all terminal servers in the cluster. When using this command, it's convenient if password-less SSH login between the servers in the clusters has been set up.

**Best Practice:** An alternative approach to using password-less login is to use the SSH agent to cache the passphrase of a SSH keypair. This increases the security, since a malicious party that gains access to the server which is configured to login to the other servers with SSH key-pair does not automatically get access to the rest of the servers - a password is needed.

First, setup the SSH key-pair as described below:

```
#
# First time / One time procedure
#
# Generate a private and public key-pair for SSH with SSH keygen.
# When prompted pick a secret password for the key-pair.
#
ssh-keygen -t dsa

# Copy the public key to SSH authorized_keys
cp /root/.ssh/id_dsa.pub /root/.ssh/authorized_keys

# Make sure the authorized key has the right permissions
chmod 600 /root/.ssh/authorized_keys

# Copy the authorized key to all Thinlinc Agents
tl-rsync-all /root/.ssh/authorized_keys
```

Next, before using tl-ssh-all, do as follows

```
eval `ssh-agent`
ssh-add

# Run your commands
tl-ssh-all rpm -Uvh /root/kdelibs-3.5.1-1.fc4.i386.rpm
```

## Commands in /opt/thinlinc/libexec

### tl-crossover-drives

CodeWeavers CrossOver allows you to configure the mapping between Windows drive letters and paths in the Linux file system. This can be done globally by adding symbolic links to the directory /opt/cxoffice/support/BOTTLENAME/dosdevices. However, this does not work if drive letters should correspond to different paths for different users. In this case, a bottle hook script is required. **tl-crossover-drives** is such a script that automatically maps "personal" mounts to separate drive letters in CrossOver. This includes all mounts mounted on subdirectories in the users home

directory. The first character of the directory name determines the drive letter. To activate this command for all bottles, execute:

```
# mkdir /opt/cxoffice/support/scripts.d
# ln -s /opt/thinlinc/libexec/tl-crossover-drives \
/opt/cxoffice/support/scripts.d/02.tl-crossover-drives
```

### **tl-has-gnome-2**

The **tl-has-gnome-2** command is used to check if Gnome 2 is installed on the system, in a way which works for most distributions. It is used by the default profile configuration.

### **tl-unity-2d** [*--test*]

The **tl-unity-2d** command is used to start the Unity 2D desktop environment, in a way that works on most distributions. It is used by the default profile configuration. The *--test* option can be used to test if this desktop environment is installed.

### **tl-kinit.sh**

The **tl-kinit.sh** command is used to obtain a Kerberos ticket automatically during start of the session, using the single sign-on mechanism.

### **tl-kdestroy.sh**

The **tl-kdestroy.sh** command is used to destroy the Kerberos ticket cache. It calls **kdestroy** during logout.



## Chapter 14. Server Configuration

### 14.1. Configuring ThinLinc Servers in a Cluster

In this section, we will describe how to configure a ThinLinc cluster with multiple agent servers to allow load-balancing and redundancy.

**Note:** This section does *not* address configuration of high availability (HA). For information on configuring your ThinLinc cluster for high availability, see Chapter 6.

A ThinLinc cluster consists of one master server (or multiple master servers in a HA configuration) with multiple agent servers behind it. While ThinLinc in its simplest configuration may be run with both the master and agent installed on the same machine, running ThinLinc in a cluster configuration conveys numerous advantages:

1. A cluster configuration allows automatic load-balancing of sessions across multiple agents
2. Having multiple agents offers redundancy; for example, if one agent goes down or is taken out of service for maintenance, other agents are still available to handle user sessions
3. A cluster configuration is scalable. Since most of the workload is taken up by the agents and not the master, adding more capacity to your ThinLinc installation is generally as easy as installing one or more new agent servers

#### 14.1.1. Configuration Options

When configuring ThinLinc servers as a cluster, there are two main configuration options which need to be set, one on the master and one on the agent(s):

`/vsmserver/terminalservers`

This parameter is configured on the master, and contains a list of all agent servers which will be part of this cluster. Only servers in this list will be considered by the master as available to create sessions on. See Section 14.2.2 for details.

`/vsmagent/master_hostname`

This parameter is configured on each agent in the cluster, and contains the name of the master server for the cluster. See Section 14.2.1 for details.

Once the two parameters above have been configured, and the `vsmagent` and `vsmserver` services have been restarted, these ThinLinc servers will then function as a cluster.

#### 14.1.2. Cluster Management

When multiple agents have been configured as part of a cluster, it is usually desirable to keep their configurations synchronised. Instead of making the same change separately on each agent, ThinLinc

ships with the tool `tl-rsync-all` to ensure that configuration changes can be synchronised easily across all agents in a cluster. See Chapter 13 for more information on how to use this tool.

The `tl-rsync-all` command should be run on the master server, since it is the master which has a list of all agents in the cluster (in `/vsmserver/terminalservers`). For this reason, it is often useful to configure the master server as an agent as well, even if it will not be used to host user sessions in general. This allows the master to be used as a "template" agent, where configuration changes can be made and tested by an administrator before pushing them out to the rest of the agents in the cluster. In this way, configuration changes are never made on the agents themselves; rather, the changes are always made on the master server, and then distributed to the agents using `tl-rsync-all`.

An example of how one might implement such a system is to configure the master server as an agent which only accepts sessions for a single administrative user. The steps to do this are as follows:

1. Configure the master as an agent too. On a ThinLinc master, the `vsmagent` service should already have been installed during the ThinLinc installation process; make sure that this service is running.
2. Make sure that the master server is *not* listed in the parameter `/vsmserver/terminalservers`. This will ensure that normal users will not be able to create sessions on the master server.
3. Create an administrative user, for example `tladmin`. Give this user administrative privileges if required, i.e. `sudo` access.
4. Make sure that the master server is explicitly selected as the agent for `tladmin` whenever they create a session. This is done using the parameter `/vsmserver/explicit_agentselection`. See Section 14.4.9 for details on this parameter. As an example, the parameter should contain the value `tladmin:master.example.com`.

In this way, configuration changes are never made on the agents themselves; rather, the changes are always made on the master server, and then tested by logging in as the `tladmin` user. If successful, these changes are then distributed to the agents using `tl-rsync-all`.

## 14.2. Server Configuration Parameters

The ThinLinc server is configured using a number of configuration parameters stored in Hiveconf. For information about how to access and set the parameters, please refer to Chapter 15. In this chapter, we will describe the different parameters and their meaning.

The parameters used in ThinLinc are divided into a number of folders, each having zero or more subfolders. The following folders exist:

- `/vsm/` contains parameters common to both the VSM agent and the VSM server. This folder normally resides in `/opt/thinlinc/etc/conf.d/vsm.hconf`
- `/vsmagent/` contains parameters specific to the VSM agent. This folder normally resides in `/opt/thinlinc/etc/conf.d/vsmagent.hconf`
- `/vsmserver/` contains parameters specific to the VSM server. This folder normally resides in `/opt/thinlinc/etc/conf.d/vsmserver.hconf`
- `/appservergroups/` contains parameters used by ThinLinc to access application servers. This folder normally resides in `/opt/thinlinc/etc/conf.d/appservergroups.hconf`

- `/profiles/` contains parameters for configuring the different session profiles. This folder normally resides in `/opt/thinlinc/etc/conf.d/profiles.hconf`
- `/utils/` contains parameters used by miscellaneous ThinLinc utilities. Each utility has its own configuration file, but all parameters are then merged in under `/utils` when read by the HiveConf framework.
- `/sessionstart/` contains some parameters used during session startup.
- `/tlwebadm/` contains parameters for the tlwebadm web configuration interface.
- `/webaccess/` contains parameters for the server part of ThinLinc HTML5 browser client.

### 14.2.1. Parameters in `/vsmagent/`

In this section, we will describe all the parameters currently used by the VSM agent.

`/vsmagent/agent_hostname`

Public hostname; the hostname that clients are redirected to. If not defined, the agent will use the computer's IP address. This is the default configuration, and means that ThinLinc does not require DNS to work properly. However, if you are using Network Address Translation (NAT), you must set this parameter to a IP address or DNS name that all clients can connect to. Example:

```
agent_hostname = thinlinc.example.com
```

`/vsmagent/allowed_clients`

This is the list of VSM servers that should be allowed to connect to this VSM agent and create new sessions. The localhost is always allowed as well as the IP of the hostname the VSM agent runs on, and the host specified in the `/vsmagent/master_hostname/` parameter.

`/vsmagent/default_environment`

This subfolder of `/vsmagent` contains environment variables that should be set in each user's session. Example:

```
[/vsmagent/default_environment]
LANG=sv_SE
LC_CTYPE=sv_SE.UTF-8
FOOBAR=foobar
```

This will set the `LANG` environment variable to `sv_SE`, the `LC_CTYPE` variable to `sv_SE.UTF-8` and the `FOOBAR` variable to `foobar` in each user's session.

**Note:** Since `xstartup` is run through `/bin/bash --login`, files in `/etc/profile.d` will be sourced and may override values in `[/vsmagent/default_environment]`

`/vsmagent/default_geometry`

The default session size, to be used when clients are not requesting any specific session size.

`/vsmagent/display_max`

The maximum display number to be used for ThinLinc sessions on each specific VSM agent host. Default value is 2000.

The maximum ThinLinc sessions allowed on a specific VSM Agent host is

`/vsmagent/display_max - /vsmagent/display_min`.

`/vsmagent/display_min`

The lowest display numbers to use for clients. The default is 1, and unless there are other processes needing display numbers, the recommendation is not to change this number. See Appendix A for an in-depth explanation of port allocation.

`/vsmagent/listen_port`

The TCP port VSM Agent listen to for incoming requests. This should normally be set to the same value as `/vsm/vsm_agent_port`.

`/vsmagent/lowest_user_port`

The lowest port to be used by normal user processes. This may *never* be lower than `/vsmagent/max_session_port`. See Appendix A for an in-depth explanation of port allocation.

`/vsmagent/make_homedir`

If this parameter is true, the users home directory will be automatically created if it doesn't exist.

`/vsmagent/make_homedir_mode`

When a home directory is created (see parameter `/vsmagent/make_homedir` above), the mode for the newly created directory will be determined by this parameter.

`/vsmagent/master_hostname`

This parameter specifies the hostname of the master machine, i.e. the machine that runs the VSM server. In a HA setup, this should be the hostname of the IP address that is on the machine that is currently the active node, to ensure that services on the agents that need to access the VSM Server always connects to the machine that is up and running.

`/vsmagent/max_session_port`

The highest port to use for VNC and tunnel ports on the VSM Agent. See Appendix A for an in-depth explanation of port allocation.

`/vsmagent/single_signon`

This parameter decides whether the passwords of the users should be saved in order to support Single Sign-On when connecting to servers from the ThinLinc session, for example when running a Windows session.

`/vsmagent/xserver_args`

Extra arguments to pass on to the Xserver Xvnc. One common case is to use `-localhost`, which makes Xvnc require connections to originate from localhost, thus forcing applications to either be local or use a tunnel (which often also means that the traffic is encrypted). Other examples include `-IdleTimeout` and `-MaxIdleTime`. For more information, see Section 14.5.



`/vsmagent/xauthority_location`

This parameter controls the location of the `Xauthority` file. Currently, two values are supported: With "homedir", the file will be placed in the users home directory. With "sessiondir", the file will be placed in the session directory below `/var/opt/thinlinc/sessions`. The `XAUTHORITY` environment variable is set accordingly by the VSM agent.

### 14.2.2. Parameters in `/vsmserver/`

In this section, we will describe all the parameters currently used by the VSM server.

`/vsmserver/admin_email`

The administrator's email address. This is where warnings about overuse of Licenses are sent, among with other administrative messages. Make sure this is a valid address.

`/vsmserver/agentcomm_bind_hostname`

The hostname the outbound requests from the VSM server to the VSM agents should come from. This parameter should be set if the machine running VSM server has several interfaces in order to make sure that the correct interface is used for outbound connections to the VSM Agents

`/vsmserver/allowed_clients`

A list of hosts from which privileged operations are allowed. The default (empty) allows localhost to do this. Privileged operations are for example to deactivate a session, something that should be allowed by the host running the ThinLinc Web Administration service.

`/vsmserver/allowed_groups`

ThinLinc access can be limited to certain groups. If `allowed_groups` is empty, all users are accepted. Otherwise, the user must be a member of the groups listed below, to be able to use ThinLinc. Example:

```
allowed_groups = students teachers
```

`/vsmserver/allowed_shadows`

A list of users that are allowed to shadow other users. Please note that these users will gain full access to other users' sessions.

`/vsmserver/explicit_agentselection`

This parameter presents a way to force the sessions created for certain users or groups to always be created on specific agent hosts. See Section 14.4.9 for more information.

`/vsmserver/terminalservers`

All ThinLinc machines part of this ThinLinc cluster. This should be a list of DNS host names. These will be used for communication between the server and the agent. The names reported to clients are fetched from the agent itself; names in `/vsmserver/terminalservers` are not reported directly to clients.

`/vsmserver/bogomips_per_user`

Estimated bogomips required for each user.

`/vsmserver/existing_users_weight`

This parameter decides the importance of the amount of logged in users on a VSM agent host when calculating load balance parameters. A host with low load, but a lot of users, is generally more likely to get a higher load within short time when the users get active. For this reason, the load balance calculating code takes the number of users at a certain host into its calculation. The `/vsmserver/existing_users_weight` controls how important this factor is. A higher value of this parameter means the load balancing code will care less about a high number of users on a certain machine.

**Note:** This parameter should normally not be changed, unless when fine-tuning the load balancing.

`/vsmserver/HA/enabled`

If this parameter is true, the VSM server will try to replicate information about sessions to the other VSM server node. See Chapter 6 for more information about ThinLinc in a High Availability configuration.

`/vsmserver/HA/nodes`

This parameter lists the hostnames of both nodes in a ThinLinc HA setup. The list should include the hostname of the current node. This means that `vsmserver.hconf` can be identical on both nodes.

`/vsmserver/listen_port`

The TCP port VSM Server listen to for incoming requests. This should normally be set to the same value as `/vsm/vsm_server_port`.

`/vsmserver/load_update_cycle`

The number of seconds allowed for updating the load status in the entire cluster.

`/vsmserver/max_sessions_per_user`

The maximum number of sessions allowed per user. 0 means no limit.

`/vsmserver/ram_per_user`

Integer, number of estimated MiB memory required for each session. A value of 8 is appropriate if only **tl-run-windesk** is used.

`/vsmserver/sshd_log_files`

A list of **sshd** log files. These files are used for determining the client IP. Different systems use different files. VSM server will try all listed files in order. If this list is empty, VSM server will not try to determine the client IP.

`/vsmserver/unbind_ports_at_login`

If this parameter is true, processes occupying the users' interval of forwarded ports will be killed at login. This means that if a user logs in twice to the same session, the second login will get working tunnel ports, if this parameter is true. The first session's tunnel ports will stop working. If the parameter is false, the first session will keep the ports.

### 14.2.3. Parameters in `/vsm/`

Parameters in the `/vsm/` folder are used by both the VSM agent and the VSM server. Neither of them need to be changed on a normal ThinLinc installation.

`/vsm/tunnel_bind_base`

The tunnels setup by the client to access various resources (audio, serial port, network resources, local printer) need one port number each on the server running the VSM agent the client is connected to. This parameter decides the lowest such port that is allocated by the VSM agent. Each user has a port range defined by the formula `/vsm/tunnel_bind_base + display-ID*10 + service_slot` where the `service_slot` depends on which service will use the tunnel. This port range is however used only for sessions with display numbers less than 100. See Appendix A for an in-depth explanation of port allocation.

**Note:** This parameter should normally not be changed.

`/vsm/tunnelservices/`

There are several parameters under the `/vsm/tunnelservices` folder. Each one decides which ports are used at serverside termination points for the tunnels used to access client resources. See Appendix A for an in-depth explanation of port allocation.

**Note:** None of these parameters should normally be changed.

`/vsm/tunnelslots_per_session`

The number of ports to reserve for tunnel port endpoints on the server. The number of ports actually used depends on the number of services defined under `/vsm/tunnelservices/`. We recommend letting this parameter have its default value (10), since that leaves for further services and easy live upgrades of ThinLinc. See Appendix A for an in-depth explanation of port allocation.

`/vsm/vnc_port_base`

The port base for VNC communication. The VNC protocol runs on one port per active user on the VSM agent host, and this is the base of the numbers used. That is, for the first user, the port will be `/vsm/vnc_port_base + 1`, for the second user `/vsm/vnc_port_base + 2` and so on. This

algorithm is used only for display numbers below 100. See Appendix A for an in-depth explanation of port allocation.

**Note:** This parameter should normally not be changed.

`/vsm/vsm_agent_port`

VSM agent communication. This is the port that the VSM server connects to on VSM Agents. This traffic is not encrypted.

**Note:** This parameter should normally not be changed

`/vsm/vsm_server_port`

The port that the VSM server listens to.

**Note:** This parameter should normally not be changed

#### 14.2.4. Parameters in `/appservergroups/`

Parameters related to how ThinLinc connects to application servers, such as UNIX servers via the X Window system, or Windows Terminal Servers using the RDP protocol, are stored under `/appservergroups/` in the Hiveconf tree. There are two subfolders of `/appservergroups/`, `rdp` and `x11`. The `rdp` subfolder is used for settings related to connections to Windows Terminal servers. The `x11` subfolder contains settings related to UNIX X11 and Linux servers. Each of the two subfolders have one or more subfolders. Each subfolder represents an application server group, a way of configuring what server a specific user should be connected to. The commands **tl-run-unixapp**, **tl-run-winapp**, **tl-run-winapp-seamless**, **tl-run-windesk**, and **tl-run-rdesktop** all take the parameter `-G` to choose which appserver group to connect to. If no `-G` parameter is given, they connect to the group named `default`.

`/appservergroups/rdp/<appgroup>/domain`

The Windows NT domain to use.

`/appservergroups/rdp/<appgroup>/keyboard_layout`

The keyboard layout to use for connections to Windows Terminal Servers. If no layout is specified, the appropriate keyboard layout will be determined automatically based on the session's locale settings.

`/appservergroups/rdp/<appgroup>/novell`

Set this parameter to `true` to improve compatibility with servers that authenticate against Novell eDirectory.

`/appservergroups/rdp/<appgroup>/rdesktop_args`

Extra arguments for RDP connections to Windows Terminal Servers. See the documentation for `tl-run-rdesktop` in Chapter 13 for information about the possible values of this parameter.

`/appservergroups/rdp/<appgroup>/redirect_printers`

True if printers should automatically be redirected to Windows Terminal Servers. See Section 5.6 for details.

`/appservergroups/rdp/<appgroup>/servers`

A list of Windows Terminal Servers to connect to using the RDP protocol. This list is read by `tl-run-rdesktop` (and associated commands) to decide which Windows Terminal Server to connect to. The Windows Terminal Server with the least load is chosen.

`/appservergroups/rdp/<appgroup>/sound`

This parameter determines the sound system to use. If set to "esddsp", sound redirection using the "esddsp" wrapper will be enabled. A value of "padsp" uses the PulseAudio (<http://pulseaudio.org/>) system instead. If "auto" is specified, "padsp" and "esddsp" are both tried, in that order. The empty string disables sound redirection.

`/appservergroups/x11/<appgroup>/servers`

A list of external UNIX servers to connect to when the `tl-run-unixapp` is called.

**Note:** In the current release of ThinLinc, load balancing is not supported when connecting to UNIX servers, so only the first server in this list will be used.

`/appservergroups/x11/<appgroup>/use_ssh_encryption`

True if X11 traffic should be encrypted via SSH.

`/appservergroups/x11/<appgroup>/xauth_path`

The path to the xauth executable on the remote server. This is only used if `use_ssh_encryption` is false.

### 14.2.5. Parameters in `/sessionstart/`

In this section, we will describe all the parameters currently used by the session startup scripts.

`/sessionstart/background_color`

The initial color of the background that is set early during session startup. By default this is a dark blue color.

`/sessionstart/background_image`

A PNG image used as the initial background. The image will always be scaled to cover the entire screen.

If the image contains transparency then the color set by `background_color` will shine through.

`/sessionstart/keyboard_layout`

The default virtual keyboard layout used by Xvnc. The protocol is not dependent on this being configured, but some applications can misbehave if a different virtual layout is configured compared to the real keyboard layout on the client device.

A list of possible keyboard layouts is given from this command:

```
$ man /opt/thinlinc/share/man/man7/xkeyboard-config.7
```

### 14.2.6. Parameters in `/tlwebadm/`

For details of parameters in `/tlwebadm/`, see Section 16.2

### 14.2.7. Parameters in `/webaccess/`

For details of parameters in `/webaccess/`, see Section 8.7.3.1.1

## 14.3. Configuring Logging on ThinLinc servers

In this section we will describe how ThinLinc logs activities on the server, and how the logging can be configured.

Logs are written by each individual session and by the following ThinLinc server components.

- VSM server
- VSM agent
- Web Integration
- Web Administration Interface
- Web Access (HTML5)

### 14.3.1. ThinLinc server components

Logging is configured by editing parameters in Hiveconf. Each component that uses the logging system has a Hiveconf folder named `logging`. In this folder and its subfolder, the logging system is configured.

#### 14.3.1.1. Log destinations

Logs can be written either to file on the local disk, to syslog or both.

#### 14.3.1.1.1. Writing Logs to File

The file name for the log file written to local disk is configured by editing the parameter `logfile` under the `logging` folder. To turn off logging to file, set the parameter `log_to_file` to 0. Note that the log file will still be created. If abnormal situations occur because of programming errors, data may appear in the file.

#### 14.3.1.1.2. Writing Logs to Syslog

For large installations, using a central loghost might be very convenient. ThinLinc supports writing logs to syslog, which makes it possible to collect all logs in one place.

By setting the parameter `log_to_syslog` under the `logging` folder to 1, logs will be written to syslog. Specify the syslog facility in the parameter `syslog_facility`. The default behaviour is not to log to syslog.

If the parameter `syslog_host` is set, logs will be sent via UDP to the syslog daemon on the host specified. If not, logs will be sent to syslog by writing to the socket specified in `syslog_socket`. The latter is the default.

### 14.3.1.2. Subloggers

Each program doing logging uses a number of sub loggers. Sub loggers are a way to distinguish different types of information written by the program. For example, the VSM server uses the subloggers *license*, *session* and *shadow* for logging license-related messages, information about sessions and information about shadowing respectively. Every sublogger can be configured to use a different log level. This allows the system administrator to, for example, increase the information about new sessions without increasing the overall loglevel, easing debugging of specific problems.

### 14.3.1.3. Log levels

The amount of logging can be configured using log levels. The log levels available are:

**Table 14-1. Log Levels**

Log Level	Explanation
ERROR	Unrecoverable Errors
WARNING	Warnings - something went wrong, but ThinLinc can recover.
INFO	Messages that are useful in daily maintenance.
DEBUG	Messages that can be of use for system administrators when debugging problems.
DEBUG2	Messages useful to trained ThinLinc personel when doing advanced debugging.

The log level configured can be seen as a barrier. If the log level is set to for example INFO, log messages with a level of INFO or higher are let through. If the log level instead is set to DEBUG2, all log messages are let through, since all log levels are higher than DEBUG2.

There is one default loglevel, and one loglevel per sublogger defined. If the log level for a sub level is set to a lower value than the default loglevel, more information will be written by that specific sublogger.

The default loglevel is configured in the `logging/defaultlevel` parameter. Each sublogger's level can then be configured by setting the parameters under the `logging/levels` folder.

#### 14.3.1.4. Summary

The default logging configuration is summarized in Table 14-2.

**Table 14-2. Default Log Behaviour**

Component	Default Behaviour	Log Configuration Hive Folder
VSM server	Log to <code>/var/log/vsmserver.log</code>	<code>/vsmserver/logging</code>
VSM agent	Log to <code>/var/log/vsmagent.log</code>	<code>/vsmagent/logging</code>
Web Administration Interface	Log to <code>/var/log/tlwebadm.log</code>	<code>/tlwebadm/logging</code>
Web Access (HTML5)	Log to <code>/var/log/tlwebaccess.log</code>	<code>/webaccess/logging</code>

#### 14.3.2. Per-Session Logging

Each Session writes what is written to standard output and standard error output to a file named `xinit.log` which is located in the session directory for a specific session. For example, the log for the last session of the user `pelle` is located in `/var/opt/thinlinc/sessions/pelle/last/`. This log contains for example output written by software run in the session, but it also has some output from ThinLinc software that is run by the user.

### 14.4. Customizing the User's Session

In this section, we will describe how the session startup in ThinLinc can be customized.

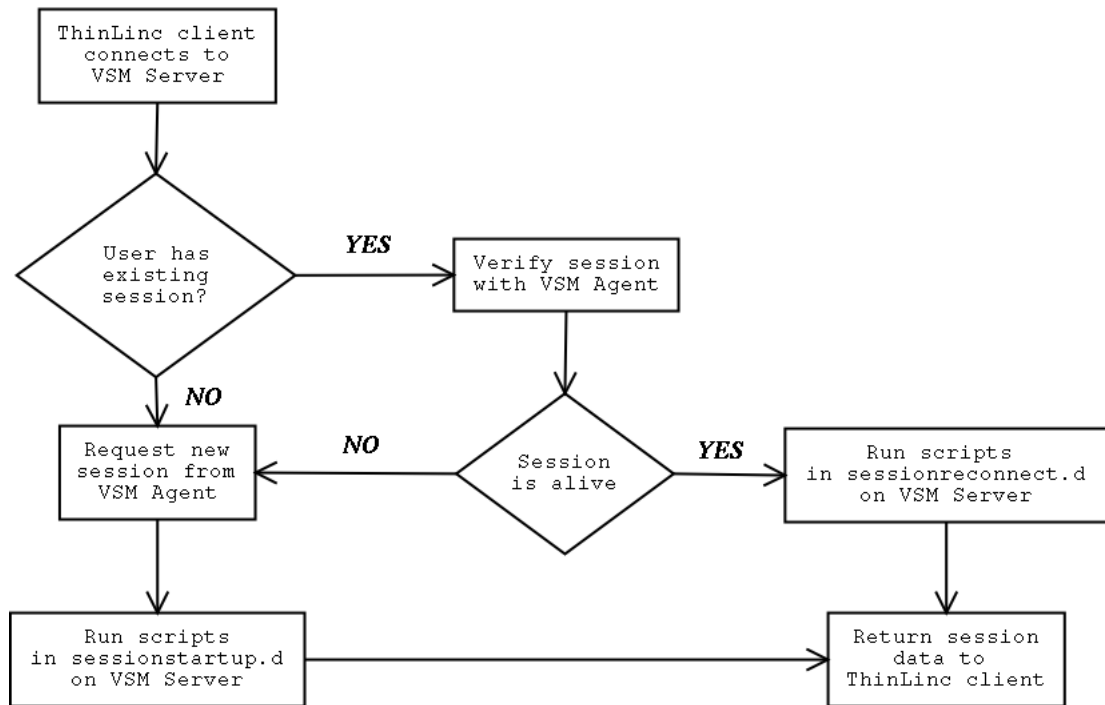
#### 14.4.1. Session startup - the big picture

The session setup is constructed to be easy to use and configure yet still easy to customize for advanced use cases.



Figure 14-1. Session Startup Procedure - on VSM Server.

## ThinLinc session startup - on VSM Server



In Figure 14-1, shows a (simplified) description of what happens on the VSM Server when a client connects to login:

- The VSM Server checks if the user has an existing session.
- If a session exists, VSM Server contacts the VSM Agent running on the host where the session is running, and asks it to verify that the session is still alive.
- If the session was alive, VSM Server runs any scripts placed in `/opt/thinlinc/etc/sessionreconnect.d`. When all such scripts are completed, session information is returned to the client. The client proceeds by contacting the agent on which the session is running.
- If the existing session was not alive, or if there were no existing session at all, VSM Server finds out which VSM Agent has the least load, and contacts this agent to request a new session.
- When the agent responds that a new session has been created, VSM Server runs any scripts placed in `/opt/thinlinc/etc/sessionstartup.d`. When all such scripts are completed, session information is sent back to the client. The client proceeds by contacting the agent on which the session was started.

#### 14.4.1.1. Scripts run at session startup/reconnect

Scripts in `/opt/thinlinc/etc/sessionstartup.d` and `/opt/thinlinc/etc/sessionreconnect.d` are run by the `root` user, on the VSM Server. Session information will not be sent back to the client until these scripts have completed. This makes it possible to ensure that commands have been run before the client connects to the VSM Agent.

If background execution is desired, place the command to be run in the background and make sure all file descriptors are closed. Here's an example on how to execute a script in the background.

```
{TLPREFIX}/sbin/tl-limit-printers < /dev/null > /dev/null 2>&1 &
```

#### 14.4.2. Session startup on VSM Agent

Figure 14-2. Session Startup Procedure - on VSM Agent

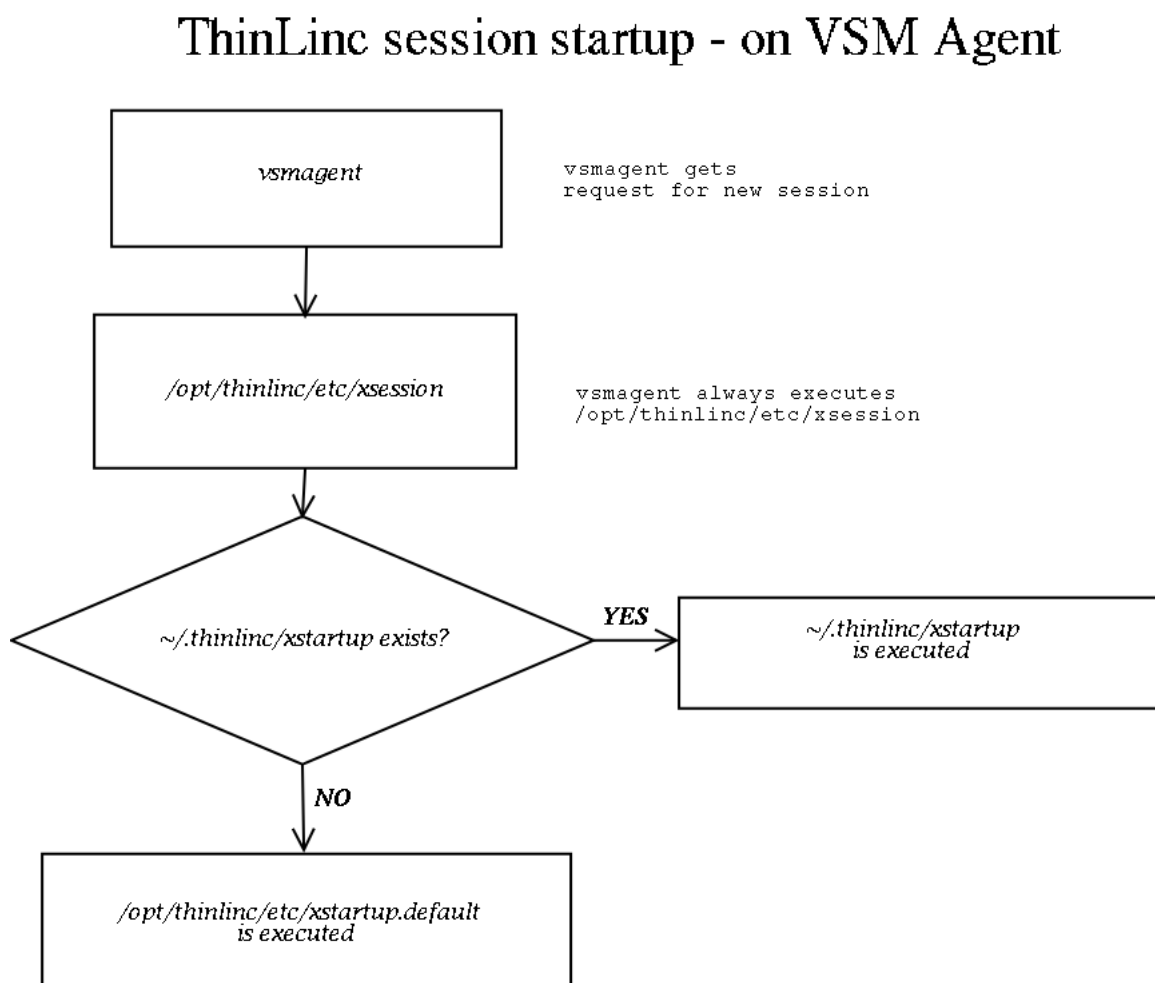


Figure 14-2 outlines what happens when an VSM Agent is contacted by VSM Server to request a new session. In detail, the following happens:

1. The VSM agent on the machine where the session will reside executes the script **/opt/thinlinc/etc/xsession**.
2. The file **/opt/thinlinc/etc/xsession** is a shell script that can be customized by advanced users. In its standard version, as delivered with ThinLinc, it will check if there is a file named **~/.thinlinc/xstartup** in the user's home directory. If there is such a file, it will be executed. If no such file exists, the file **/opt/thinlinc/etc/xstartup.default** is executed instead. See Section 14.4.3 for a description of the standard behaviour of this file.

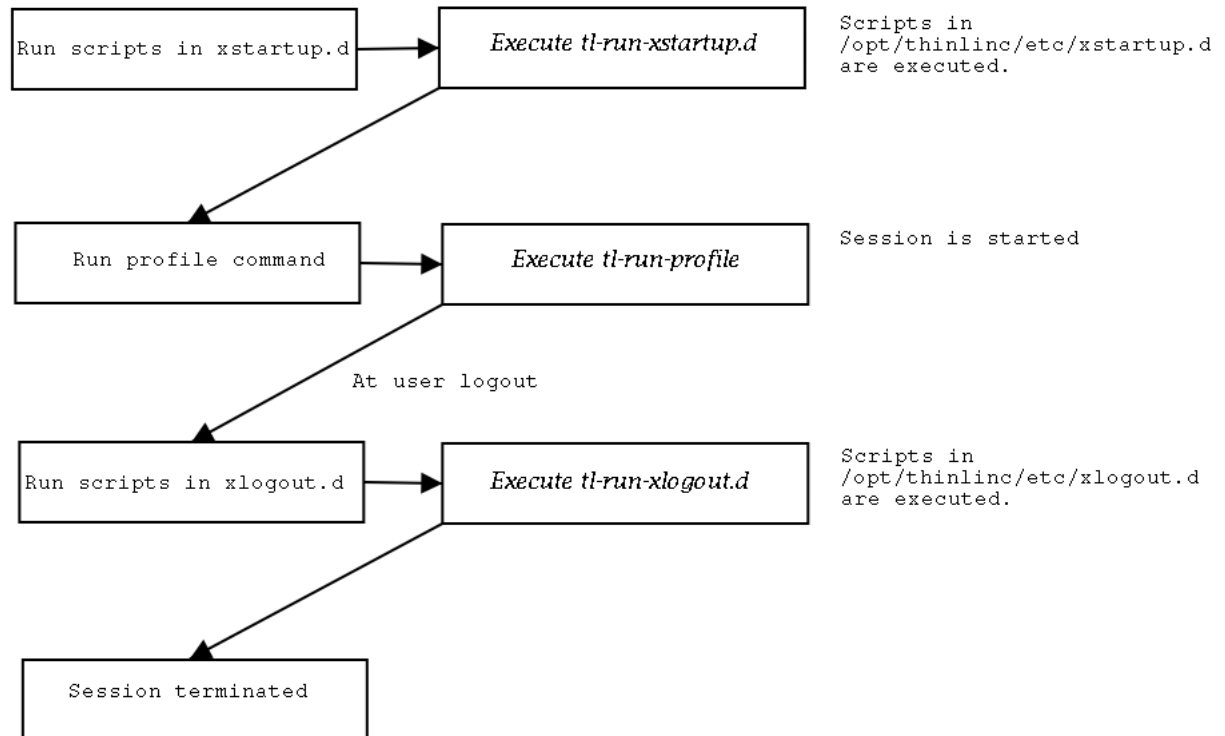
This system allows for experienced users to customize how their session startup should work by editing the file **~/.thinlinc/xstartup**. On the other hand, at sites where users should not be able to customize their system startup, **/opt/thinlinc/etc/xsession** can be modified so that it doesn't try to execute user-specific xstartup-files. The standard setup should however suit the needs of the majority of installations.

### 14.4.3. Profiles and the standard xstartup.default file.

ThinLinc allows for different "profiles" when starting up a user session. The users will be presented with a menu after logging in, where they can choose for example between a desktop suited for engineering users, a desktop suited for the marketing department or a Windows desktop. The example configuration files that are delivered with ThinLinc have several different alternatives, however only those sessions that are actually available on the system are displayed. This is just an example configuration, meant to be customized for each individual ThinLinc installation.

Figure 14-3. The ThinLinc profiles and xstartup.default

## ThinLinc profiles and xstartup.default



As described in Section 14.4.1, `/opt/thinlinc/etc/xstartup.default` is executed if there is no `~/.thinlinc/xstartup` for the user. This file, in its unmodified version as delivered with ThinLinc, executes three steps, as outlined in Figure 14-3.

1. The command **`tl-run-xstartup.d`** is executed, which causes all files in `/opt/thinlinc/etc/xstartup.d/` to be executed. Files that have filenames ending with `.sh` will be *sourced* as shell scripts. Other files are executed normally. This way, environment variables that persist down to the session command can be set in `*.sh` files.

If a specific execution order is needed for the scripts in the `xstartup.d/` directory, let the names of the scripts begin with numbers, where a script with a lower number will be executed before one with a higher number. For example `10setuphomedir` will be executed before `20copyfiles`.

By default, the script `/opt/thinlinc/etc/xstartup.d/20-tl-select-profile.sh` will invoke **`tl-select-profile`**, to let the user choose among the possible profiles. See Section 14.4.5 for documentation on how to setup profiles. If only one profile is available, **`tl-select-profile`** will select it without asking the user. The environment variable `TLPROFILE` is set to the name of the selected profile.

Worth noting is that the environment variable `TLPROFILE` is available when running the scripts in `xstartup.d`, for decisions based on what profile will be run.

2. The command **tl-run-profile** is run. This runs the commands associated with the selected profile, for example **startkde** to start a KDE session.
3. When the commands run by **tl-run-profile** exits, **xstartup.default** runs the command **tl-run-xlogout.d** which runs scripts and commands located in the directory `/opt/thinlinc/etc/xlogout.d`. The same information that applies to files in `xstartup.d` (as documented in 1) applies to files in this directory.

**Note:** Scripts in `/opt/thinlinc/etc/xstartup.d` and `/opt/thinlinc/etc/xlogout.d` are run *on the agent*, with the same rights as the user owning the session.

#### 14.4.4. Session Startup with a Client Supplied Start Program

If the client has requested that the session should be started with a command supplied by the client, VSM agent will set the environment variable `TLCOMMAND` to this command. In this case, the profile selection dialog will be disabled and **tl-run-profile** will execute the command specified by the client, instead of a profile command. To disable client supplied start programs, create the file `/opt/thinlinc/etc/xstartup.d/00-no-startprog.sh`, containing:

```
unset TLCOMMAND
```

#### 14.4.5. Configuring available profiles

The profiles choosable via the **tl-select-profile** command are configured via Hiveconf, under the `/profiles` path. The default configuration includes a number of examples.

The `default` parameter must be present, and specifies the default profile. The profile chooser will have this entry selected when it starts, and it may also be used automatically for some error conditions.

The `order` parameter selects which profiles should be available for selection, and the order in which they are displayed.

If the `show_intro` parameter is true, a configurable introduction text will be displayed and requires user input to proceed with the logon process. The `introduction` parameter is a text that will be displayed if introduction is shown, this text block does also supports Pango Markup format styling for a fancier text layout.

Each profile is defined under a section named `/profiles/<profile key>`. It has the following fields:

##### `xdg_session`

Connects this ThinLinc profile with a system desktop session configuration. The directories `/etc/X11/sessions` and `/usr/share/xsessions` will be searched for a file matching `<xdg_session>.desktop`. It is recommended that this field is used for all modern desktop environments as it sets up important environment variables.

The fields *name*, *description*, *icon*, *cmdline* and *testcmd* will all be implicitly filled in by the system configuration. You can override those values individually by specifying a different value in the ThinLinc configuration.

Multiple values can be specified in this field. The first matching configuration will be used. If no matching configuration can be found then the profile will not be shown.

**Note:** If the configuration is listed in `/etc/upstart-xsessions` then the specified command is ignored and an Upstart user session will be started instead. A manually specified *cmdline* can still be used to override the command.

**name**

A short description of the profile, shown in the profile list.

**description**

A longer description, shown under the screen shot when the profile is selected.

**icon**

A 22x22 image shown next to the name in the profile list. Paths can be absolute or relative  
`/opt/thinlinc/share/tl-select-profile.`

**screenshot**

A 200x150 image shown when the profile is selected. Paths can be absolute or relative  
`/opt/thinlinc/share/tl-select-profile.`

**cmdline**

The command to execute if this profile has been chosen.

If *xdg\_session* is set then the environment variable `XDG_EXEC` will be set to the original command line from the system desktop session configuration.

**testcmd**

A shell expression or command that is executed to determine if this profile should be visible or not. A non-zero return code causes the entry to be hidden. If this field is empty or missing then the entry will always be shown.

If *xdg\_session* is set then the environment variable `XDG_TRY_EXEC` will be set to the expected binary from the system desktop session configuration. Note that this value differs in behaviour from *testcmd*. `XDG_TRY_EXEC` should only name a executable binary in `PATH`, whilst *testcmd* will be executed and its return code inspected.

The *name* and *description* variables mentioned above also support Pango Markup (<http://developer.gnome.org/pango/stable/PangoMarkupFormat.html>) format styling which provides a simple way to display formatted text (for example, bold or italicized words).

The following example changes the name to be displayed using a blue foreground color and a large font, and the description with a boldface tag to emphasise the machine's operating system type.

```

/profiles/windows_desktop
name=<span foreground="blue" size="x-large">Windows Desktop</span>
description=This is a standard <b>Windows</b> desktop.
cmdline=${TLPREFIX}/bin/tl-run-windesk -G windows_desktop

```

#### 14.4.6. Configuring different Linux Desktops based on the selected profile

Please read Chapter 17 for documentation on how to configure different desktops with for example different menu and desktop icons depending on what profile were selected.

#### 14.4.7. Speeding up Session Startup

If a user has a complicated session startup with many time-consuming operations, it can take quite a while before the user's desktop environment (for example KDE or Gnome) begins to start. Prime examples of when this happens is when mounting local drives, or when mounting some shared directories from a Netware server.

One way of speeding up this process is to execute some of the operations in the background. Most often, there is no need to mount the local drives before starting KDE, because it takes longer time to start KDE than it takes to mount the local drives. The two operations can easily run in parallel. The same goes for the example of mounting shared directories.

The easiest way to accomplish this is to add an & sign after commands run by scripts in `/opt/thinlinc/etc/xstartup.d`.

Make sure that commands that must be run before starting the window environment are run sequentially. For example, configuring desktops via TLDC must be done before starting KDE.

#### 14.4.8. Configuring the language environment on the server based on the client language

The ThinLinc client reports the language settings on the client side when requesting a session. This can be used to configure the language on the server side. The idea is that in an environment where several languages are in use, a user could automatically get their preferred language based on what their client computer is configured for.

To activate this, a symlink needs to be created:

```
# ln -s /opt/thinlinc/libexec/tl-set-clientlang.sh /opt/thinlinc/etc/xstartup.d/00-tl-set-clientlang.sh
```

Also, make sure no other parts of the startup environment are trying to set the LANG variable. For example, on Fedora, the files `/etc/profile.d/lang.sh` and `/etc/profile/lang.csh` will override the LANG variable set by `tl-set-clientlang.sh`.

### 14.4.9. Forcing sessions for some users to certain agent hosts

In some situations, it is desirable to force sessions for certain users to be started on a specific agent host. Examples of when this is needed is when testing a new server platform, allowing a group of test users to run their sessions on the new platform, and when configuring desktops with the ThinLinc Desktop Customizer (as described in Chapter 17), where you want to end up on the same server every time to make it easier to copy the resulting files to all other hosts.

ThinLinc provides a mechanism for this. By creating a unix group and associating it with a specific agent server, sessions for users that are members of the group will always be created on the agent host in question. Individual users may also be specified in this way.

The configuration parameter used for this is `/vsmserver/explicit_agentselection`. Add pairs of user/group and agent hostnames as a space-separated list to this parameter. The names of groups should be prepended by a `+` sign, to identify them as groups.

An example:

```
/vsmserver/explicit_agentselection = +group1:agent1 +group2:agent3 user1:agent1
```

Since only one server can be associated with each group, no load balancing is used. That means that if a user that is a member of `group1` requests a session, and `agent1` is down, no session will be created.

If a server associated with a group is also listed in `/vsmserver/terminalservers`, sessions will be created for all users, not only the ones that are members of the group associated with the server. If the server is not listed in `/vsmserver/terminalservers`, only users in the group associated with the server will have sessions on the server.

### 14.4.10. Indicating that Shadowing is in Progress

At some sites it is a legal requirement that the user must be told when shadowing is in progress. By running the **tl-shadow-notify** program during the session lifetime, a window will pop up with information about the shadowing whenever shadowing starts or stops.

To activate **tl-shadow-notify**, a symlink must be created:

```
# ln -s /opt/thinlinc/bin/tl-shadow-notify /opt/thinlinc/etc/xstartup.d/15-tl-shadow-notify
```

## 14.5. Limiting Lifetime of ThinLinc Sessions

The Xserver has three options which controls the maximum lifetime of ThinLinc sessions. These are described below, and can be added to the parameter `/vsmagent/xserver_args`.

- `-MaxDisconnectionTime s`

Terminate when no client has been connected for `s` seconds. Note: Never use a value smaller than 60.

- `-MaxConnectionTime s`

Terminate when a client has been connected for `s` seconds

- `-MaxIdleTime s`



Terminate after *s* seconds of user inactivity. Note: Never use a value smaller than 60.

In addition to the options above which control the lifetime of the ThinLinc session, the option `-IdleTimeout` can be used to configure how long an idle session should remain connected. The `-IdleTimeout` option takes a number of seconds as an argument, and can be added to the parameter `/vsmagent/xserver_args` as per the options described above.

**Note:** Setting `-IdleTimeout s` will simply disconnect the client from the session after *s* seconds; it will not terminate the ThinLinc session itself.

## 14.6. Restricting SSH Daemon Port Forwarding

ThinLinc requires that the SSH daemon allows the user to create a forwarded TCP connection to Xvnc process, in order to be able to connect to the session. The default configuration of the SSH daemon is typically to allow forwarding to any destination. If you are using the OpenSSH server, version 4.5 or later, it is possible to use a more restricted configuration. To activate this, add the lines below to `/etc/ssh/sshd_config`:

```
PermitOpen 127.0.0.1:22
# @thinlinc-begin@
```

**Note:** The first line essentially turns off TCP forwarding. The "none" option can also be used if you are using OpenSSH 6.1 or later.

In this case, ThinLinc will automatically add "PermitOpen" lines between this marker and the end marker:

```
# @thinlinc-end@
```

Please note that this feature only works with an OpenSSH (or compatible) daemon. Additionally, if shadowing should be allowed this must be manually specified. For example, to exclude root from port forwarding restrictions and allow shadowing, add this to the configuration:

```
Match User root
    PermitOpen any
```



## Chapter 15. Hiveconf

### 15.1. Overview

Hiveconf is the name of the configuration system used in ThinLinc. It is however not a ThinLinc-specific configuration system, but instead a generic configuration framework for storing key/value pairs in a human readable way, although still in a format that's easy to read and modify from a computer program.

Hiveconf stores data using a "backend", meaning configuration data can be stored in different ways. The default backend which is also used in ThinLinc is using a text file format similar to Windows `.INI`-files, or the format used in `smb.conf` from Samba.

In this section, we will describe Hiveconf from a general point of view and also describe ThinLinc-specific details.

#### 15.1.1. Basic Syntax

Basically, a Hiveconf file consists of key/value pairs with a equalsign(=) between them, as in the following example:

```
vsm_server_port = 9000
vnc_port_base = 5900
```

The values after the equal sign can be of the following types:

- String
- Boolean
- Integer
- Float
- Binary data as hexadecimal ASCII

Data can also be lists of the above types.

#### 15.1.2. Tree Structure

Parameters in Hiveconf all reside in folders. Folders are just like a directory or folder in a normal file system. By adding folder directives to Hiveconf files, the parameters will be split up in a tree structure, meaning each parameter will be addressed using a path. This way, two folders can have two parameters with the same name without collision.

The benefits of this is that a software suite (for instance ThinLinc) can have one common configuration namespace, without having to name all configuration parameters uniquely, since every component in the suite can have its own namespace. In ThinLinc, the VSM server has its parameters in the `vsmserver/` folder, the VSM agent has its parameters in the `vsmagent/` folder and so on.

Looking from a system global point of view, every software package has its own folder, meaning *all* configuration parameters of the system can be accessed using a common tool.

Folders are put into the configuration files by adding a path inside square brackets to the file as in the following example:

```
[root@tlha-master conf.d]# cat vsmserver.hconf
#
# Hiveconf configuration file - VSM server
#
[/vsmserver]
unbind_ports_at_login=true

# Administrators email
admin_email = root@localhost

#
# Terminal servers. A list of hostnames. These will be used ...
# between the server and the agent. The names reported to ...
# the agent itself; names in terminalservers are not reported ...
#
terminalservers = tlagent-0.example.com
```

In this example, all three parameters ( *unbind\_ports\_at\_login* , *admin\_email* , and *terminalservers* ) reside in the */vsmserver* folder. This means that they should be addressed as */vsmserver/unbind\_ports\_at\_login* , */vsmserver/admin\_email* and */vsmserver/terminalservers* respectively if used from inside a program using the Hivetool libraries. This is of course not that important from the system administrator's point of view, but it's important to understand the principle.

### 15.1.3. Mounting Datasources

One Hiveconf file can use another Hiveconf file by mounting the other file using a mount command as in the following example:

```
%mount HA.hconf
```

The mount should be compared to a mount on a UNIX or Linux system. That is, the mount adds the tree structure of the file mounted at exactly the place in the current tree structure where the mount command was found.

Mounts can also use wildcards, as in the following example

```
%mount conf.d/*.hconf
```

The above is exactly what you'll find if you look into the file */opt/thinlinc/etc/thinlinc.hconf*. Hiveconf will mount all files in */opt/thinlinc/etc/conf.d* and add them to the current folder. This is a very convenient way to add all configuration files for a specific software suite to the Hiveconf namespace.

### 15.1.4. Hostwide Configuration

As we hinted in Section 15.1.2, Hiveconf lays the foundation for a hostwide configuration system where all applications on a host can be configured using a single system with a common API. This can be accomplished because each application will get its own subfolder in the hostwide configuration folder, so that two applications parameters won't collide even if they have the same name. Using the mount command, every application can have its own configuration file, while still exporting its parameters to the hostwide folder system.

There is a hostwide Hiveconf "root", implemented by the file `/etc/root.hconf`. This file mounts all files in `/etc/hiveconf.d/` where an application can drop its own hiveconf file at install-time, just like it can drop a file in for example `/etc/logrotate.d` or `/etc/profile.d`. There is such a file for ThinLinc that mounts the file `/opt/thinlinc/etc/thinlinc.hconf` under the folder `thinlinc`.

### 15.1.5. Hiveconf Tools

In addition to the system libraries used by applications to read and write configuration parameters that reside in Hiveconf files, there is a command line utility named **hivetool** for inspecting and setting parameters from the command line. This can be very convenient, for example when scripting setup scripts that need to set some parameter.

**Hivetool** without parameters will do nothing. To see all parameters on the system, run:

```
# hivetool -Ra /
```

which instructs **hivetool** to print all parameters, beginning from the root (`/`) and recursing downwards. With a standard Hiveconf installation this will list Samba and KDE configuration parameters. If ThinLinc is installed, it will list ThinLinc parameters as well.

To print a specific parameter, run **hivetool** with the name of the parameter as parameter. For example:

```
[root@tlha-primary etc]# hivetool /thinlinc/vmsserver/admin_email
root@localhost
```

Setting a parameter is equally easy. To set the `admin_email` parameter above, execute the following:

```
# hivetool /thinlinc/vmsserver/admin_email=test@example.com
```

## 15.2. Hiveconf and ThinLinc

ThinLinc uses Hiveconf as its primary configuration system on the serverside. In this section, we will describe the convenience utility shipped with ThinLinc. For descriptions of the folders and parameters used by ThinLinc, please refer to Chapter 14

### 15.2.1. The ThinLinc Configuration Tool - tl-config

In order to access the ThinLinc part of the Hiveconf configuration namespace without having to address it using the hostwide path (i.e. to avoid having to add `/thinlinc/` to all parameters, a tool named **tl-config** is shipped with ThinLinc.

**tl-config** takes the same parameters as **hivetool** and works the same way. Refer to Section 15.1.5 for information about **hivetool**. Try for example

```
# tl-config -Ra  
/
```

a command that will print all ThinLinc-related parameters.

## Chapter 16. Administration of ThinLinc using the Web Administration Interface

### 16.1. Introduction

This chapter describes the web-based ThinLinc administration interface called `tlwebadm`. This administration interface is installed automatically by the ThinLinc installation program, and may be accessed by pointing your web browser to `https://<hostname>:1010`. For information on configuring `tlwebadm`, for example setting a password or changing the default port, see the following section Section 16.2.

### 16.2. Configuring `tlwebadm`

`tlwebadm` offers a number of configuration options, which are stored in the configuration file `/opt/thinlinc/etc/conf.d/tlwebadm.hconf`. The various configuration options are described below.

**Note:** The password must be set and the `tlwebadm` service restarted before use.

`/tlwebadm/username`

The username to authenticate with when accessing the web interface.

`/tlwebadm/password`

The password for the above user. The tool `/opt/thinlinc/sbin/tl-gen-auth` may be used to create hashes of the format required for use with this parameter.

`/tlwebadm/cert`

The path to the certificate file to be used for TLS encryption.

`/tlwebadm/certkey`

The path to the certificate private key file.

`/tlwebadm/listen_port`

The local port for the web server to listen on.

`/tlwebadm/logging/logfile`

The file to use for logging `tlwebadm` messages. By default, this is `/var/log/tlwebadm.log`.

## 16.3. Modules

The tlwebadm interface consists of several modules which address different aspects of ThinLinc configuration:

- *System Health*, for viewing information about ThinLinc master and agent services, and testing user or group lookup performance. See Section 16.3.1 below.
- *Status*, for viewing information such as license usage, server load and sessions. See Section 16.3.2 below.
- *VSM*, for viewing information and managing ThinLinc master and agent services. See Section 16.3.3 below.
- *Profiles*, for viewing and configuring profiles. See Section 16.3.4 below.
- *Locations*, for viewing and configuring printers and terminal locations. See Section 16.3.5 below.
- *Desktop Customizer*, for configuring desktops. See Section 16.3.6 below.
- *Application Servers*, for viewing and configuring application servers.
- *Novell Configurator*, for configuring, optimising and troubleshooting Novell LDAP servers. See Section 16.3.8 below.
- *Documentation Center*, a module containing documentation and other useful information.

These modules are described in more detail in the following sections.

### 16.3.1. The System Health Module

The System Health module allows you to check the running state of the ThinLinc services VSM Master and VSM Agent. There is also a tool to perform a user or group lookup which reports the time for the lookup.

- *VSM Master* reports current running state of VSM Master service.
- *VSM Agent* reports current running state of VSM Agent service.
- *Users and Group Lookup* allows you to test performance of user and group lookup on the system.

Fill in values for **username** and/or **group** and click the **Test user and group lookup** button to perform a lookup.

### 16.3.2. The Status Module

The Status module allows you to view or manipulate the following aspects of ThinLinc, by selecting the relevant submenu:

- *Licenses* allows you to view current and historic license usage, as well as the current number of licenses.



- *Load* allows you to check the current server load on both ThinLinc and application servers.
- *Sessions* allows you to terminate, shadow or view details of sessions. This feature is described in more detail in the next section, Section 16.3.2.1

### 16.3.2.1. The Sessions Menu

When you select the sessions menu, a table with all currently active users is displayed. To perform additional tasks, click on the corresponding user name. This will bring up the session details page, which displays all the session parameters for each session the user has running. The information table is described below.

- **Terminal Server** : The DNS host name of the server that is hosting this session. If you only have one ThinLinc server, this server will host all sessions. If you have several ThinLinc servers in a cluster, new sessions will be created on the server with the lightest load.
- **Display Number** : Each session on a certain server has a unique number, the X Window System display number. Display zero is reserved, and never used for ThinLinc sessions.
- **Terminal ID** : An identification of the thin terminal. This is the terminal's ethernet hardware address (MAC address).
- **Framebuffer Size** : The size (resolution) of the active session.
- **Local Screen Size** : The size (resolution) of the terminal's screen.
- **Session process ID** : The PID (process identification number) of the tl-session process, which is the parent for all processes belonging to a certain session.
- **Command** : The command line that was specified when starting this session. This is usually empty for full desktop sessions.

Below each table, there are two buttons:

- **Terminate Session** : By clicking this button, you can terminate a session immediately. Caution: This can lead to data loss, since applications running on the ThinLinc servers may not hold unsaved data.
- **Shadow Session** : This button will generate a ThinLinc "launch file" (see Section 8.7.1) that starts the native ThinLinc client, preconfigured to shadow the current user.

**Note:** The user will not be informed that shadowing is in progress, unless `tl-shadow-notify` is enabled.

### 16.3.3. The VSM Module

VSM contains information about VSM Master and Agent services. This module allows you to start or stop the services, and modify a subset of the configuration options.

- *Home* allows you to view current status of VSM services.
- *VSM Master* allows you to start or stop the service, and modify a subset of the configuration options.
- *VSM Agent* allows you to start or stop the service, and modify a subset of the configuration options.

### 16.3.3.1. Home

On this page, you can check the status of VSM services.

### 16.3.3.2. VSM Master

On this page you can start or stop VSM Master service. There are also a subset of configuration options available for configuration of the service.

- *Service Status* gives you the ability to start or stop the VSM Master service.
- *Sessions per user* allows you to configure how many session are allowed per user.  
A value of zero means no limit and will give unlimited sessions per user.
- *Allowed Groups* allows you to configure which groups should be given access to connect to ThinLinc.  
If no groups are specified, all users will have access to connect to ThinLinc
- *Allowed Shadows* allows you to configure which users should be able to shadow other ThinLinc sessions.

Click the **Save** button when you want to save your changes to the configuration files.

**Note:** You need to restart the service to apply your changes.

### 16.3.3.3. VSM Agent

On this page you can start or stop the VSM Master service. There are also a subset of configuration options available for configuration of the service.

- *Service Status* gives you the ability to start or stop the VSM Agent service.
- *Agent Hostname* allows you to configure the hostname that clients are redirected to.

**Note:** This configuration is needed if running ThinLinc in a NAT environment. See Section 3.3.4 for more information.

- *Extra Arguments to X Server* allows you to configure additional arguments to the Xserver (Xvnc) for new sessions that are started.

Click the **Save** button when you want to save your changes to the configuration files.

**Note:** You need to restart the service to apply your changes.

### 16.3.4. The Profiles Module

On this page you can modify text shown in the profile chooser, and manage profiles. You can create or delete a profile and configure the profile order.

- *Introduction Texts* allows you to modify and manage translation of texts used in the profile chooser.
- *Profile List* allows you to configure the available profiles and their order.

#### 16.3.4.1. Introduction Texts

Introduction texts contains translation tables for greetings and introduction texts. There is also a configuration option to enable or disable the use of introduction texts.

- *Greeting Text* the text to show at the top of the profile chooser.
- *Show Introduction* disable or enable the introduction text which is shown to the user before the profile selection dialog.
- *Introduction Text* the text to show before presenting the list of profiles.

To add a new translation, fill in language code and the translated string. Click the **Save** button to save the new translation and add it to the translation table.

To delete a translation select the row using the checkbox in *Delete* column of the translation table. Click the **Save** button to carry out the actual deletion of selected rows.

#### 16.3.4.2. Profile List

The Profile List module contains functionality to manage your profiles. You can change the default profile, or create new and edit existing profiles. You can also change the order of profiles.

- *Default Profile* allows you to specify the default profile to be selected in the profile chooser.
- *Profile List* allows you to modify profiles and their order, or create new profiles.

Create a new profile by clicking the **Add new profile** button. If you want to edit an existing profile, click the profile name in the table of available profiles.

When creating a new or editing an existing profile a form is displayed. This form is shared between both modes and each field details are described below.

- **Identification**

An unique string identifier for the profile which is used when referencing this profile.

- **XDG Session Desktop**

The system desktop session configuration that this profile should be connected to.

- **Default Name**

A name for the profile which is displayed in the profile chooser.

- **Availability**

This will make the profile available to be selected and used. If you uncheck it will not be shown to the user in the profile chooser.

- **Take Description From**

The description is shown in the profile chooser when a profile is selected. This field can be a static text which is defined in the input field **Default Description**.

- **Test command** : This will take and use the output of defined **Test Command** as description for the profile.
- **Manually defined text below** : This will use the text defined in the **Default Description** field below.

- **Default Description**

A text used as description for the profile. This is text is used if **Take Description From** above is selected to use the manually defined text.

- **Icon Path**

A filename of the icon to use in the profile chooser.

- **Screenshot Path**

A filename of the screenshot to use in the profile chooser.

- **Command Line**

This command is used to start up a session, it might be a simple **xfce4-session** or one of shipped binaries like **\${TLPREFIX}/bin/tl-run-windesk -G wts\_farm**.

- **Test Command**

This command is evaluated and if it returns true, the profile is shown to the user. If the command evaluates as false, the profile will not be shown in the list of available profiles for the user.

ThinLinc includes the tool "tl-memberof-group" which may be used to test membership of groups. You can use this tool as test command, such as **\${TLPREFIX}/bin/tl-memberof-group my\_profile\_access\_group**. This example will give members of group **my\_profile\_access\_group** access to the profile.

If you only want to give a specific user access to the profile you may specify **test \${USER} = user**.

When you have filled out the form, or changed any fields, click **Save** button at the bottom of the form to save your changes into the configuration file.

To delete a profile click the profile name in table of available profiles. Then click the checkbox at the bottom of the form to verify your intention about deletion of the profile. Complete the deletion by clicking the **Delete** button.

## 16.3.5. The Locations Module

Locations contains information about locations where terminals and printers reside. A location can be a room, a floor, a house or some other type of geographical delimitation.

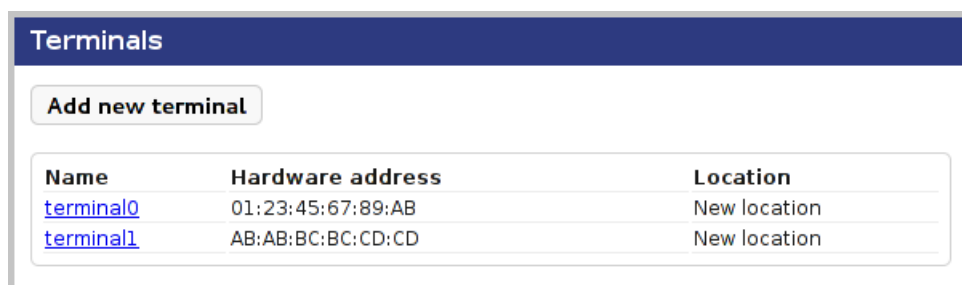
Every terminal should be assigned as a member of a location. In addition to terminals, printers may also be assigned to locations.

### 16.3.5.1. Terminals

Terminals contains necessary information about all terminals. The most important information here is every terminal's interface hardware (MAC) address.

Each terminal should be entered as described in this section. Enter the terminals module by clicking on the menu item. You will be presented with a list of currently entered terminals. This could be something like the example in Figure 16-1.

Figure 16-1. Terminals



The screenshot shows a web interface titled "Terminals" in a dark blue header. Below the header is a button labeled "Add new terminal". Underneath the button is a table with three columns: "Name", "Hardware address", and "Location". The table contains two rows of data. The first row has "terminal0" as the name, "01:23:45:67:89:AB" as the hardware address, and "New location" as the location. The second row has "terminal1" as the name, "AB:AB:BC:BC:CD:CD" as the hardware address, and "New location" as the location. The names "terminal0" and "terminal1" are underlined and appear to be clickable links.

Name	Hardware address	Location
<a href="#">terminal0</a>	01:23:45:67:89:AB	New location
<a href="#">terminal1</a>	AB:AB:BC:BC:CD:CD	New location

Figure 16-1 shows a system with a total of two terminals.

To edit a terminal, click on its name in the list.

To add a new terminal to the list you press the **Add new terminal** button. This will bring up an empty terminal edit form. See Figure 16-2 for an example.

Figure 16-2. New terminal form

The screenshot shows a web interface titled "Terminals". At the top, there is a button labeled "Add new terminal". Below this is a table with three columns: "Name", "Hardware address", and "Location". The table contains two entries: "terminal0" with hardware address "01:23:45:67:89:AB" and location "New location", and "terminal1" with hardware address "AB:AB:BC:BC:CD:CD" and location "New location". Below the table is a form for adding a new terminal. The form has four main sections: "Terminal name (required)" with a text input field containing "New terminal"; "Hardware (MAC) address (required)" with an empty text input field; "Location" with a dropdown menu showing "New location"; and "Printers" with an "Add printer" button. At the bottom of the form are two buttons: "Delete" and "Save". Below the "Delete" button is a checkbox labeled "Yes, really delete New terminal".

There are three editable fields in this view, Hardware (MAC) address, Terminal name and Location. They are described in Table 16-1 below.

To save changes, press the **Save** button. When you have pressed the **Save** button you will see that the Hardware (MAC) address field will change from being an editable field to become a static text label. To assure data integrity between the modules you aren't allowed to change this field after it's added.

When a new terminal is saved or when clicking an existing in the terminals list, there will be three buttons inside the form. The **Save** button is used to save changes made to the terminal. The **Delete** button deletes the currently viewed terminal. The **Add Printer** button will add a new printer field to the form.

Table 16-1. Terminal properties

Name	Description
Hardware (MAC) address	hardware (MAC) address of the main interface of the terminal. This field is important! Without a correct value the <i>nearest printer</i> won't work!
Terminal name	Name of the terminal. Could for example be the terminal's DNS name or a name following a naming scheme that identifies the terminal.

Name	Description
Location	Which of the locations, entered in the Locations module, this terminal belongs to.

It is also possible to add a printer to a terminal in the terminal module. This can be used if a terminal has its own printer or is closer to another printer than the ones assigned to this terminal's location. You should use this feature moderately since it may cause more administration.

To add a printer you do exactly as in the Locations menu. Click the **Add printer** button, select the printer in the pop-up menu and then press **Save** to make sure that the settings are stored. To delete it, check the relevant Delete checkbox(es) for the printer(s) you wish to remove, and click **Save**.

### 16.3.5.2. Locations

To edit a location, click on its name in the list.

To add a new location to the list you press the **Add location** button. This will bring up an empty location edit form. See Figure 16-3 for an example.

**Figure 16-3. New Location Form**

The screenshot shows a web interface titled "Locations". At the top, there is a button labeled "Add new location". Below this is a table with three columns: "Name", "Description", and "Use for Unknown Terminals". The table contains one row with the text "New location" under "Name" and "No" under "Use for Unknown Terminals". Below the table is a form for editing the selected location. This form has several sections: "Location name (required)" with a text input field containing "New Location"; "Description" with an empty text input field; "Unknown Terminals" with a checkbox labeled "Use this location for unknown terminals"; "Printers" with an "Add printer" button; and a "Delete" section with a "Delete" button and a checkbox labeled "Yes, really delete this location". A "Save" button is also present at the bottom right of the form.

Fill the Name and Description fields with relevant information. Check the checkbox if this location is to be used for unknown terminals when using the printer access control feature (see Section 5.5 for details).

To save changes, press the **Save** button. When you have pressed the **Save** button you will see that the **Name** field will change from being an editable field to become a static text label. To assure data integrity between the modules you aren't allowed to change the name of an item after it's added.

The **Delete** button deletes the currently viewed location, but only if the confirmation checkbox is also checked. The **Add Printer** button will add a new field to the form, a drop-down menu with all possible printers. An example of this can be seen in Figure 16-4.

**Figure 16-4. Location Details With Printer**

The screenshot shows a web interface titled "Locations". Below the title is a button labeled "Add new location". Below that is a table with three columns: "Name", "Description", and "Use for Unknown Terminals". The first row of the table has the values "New location", "", and "No". Below the table is a form for adding a new location. The form has three main sections: "Location name (required)" with a text input field containing "New location"; "Description" with a text input field; and "Unknown Terminals" with a checkbox labeled "Use this location for unknown terminals". Below these is a "Printers" section with an "Add printer" button and a dropdown menu showing "MX-2700N". To the right of the dropdown is a checkbox labeled "Delete". At the bottom of the form are two buttons: "Delete" and "Save". Below the "Delete" button is a checkbox labeled "Yes, really delete this location".

The **Printer** field above has the printer *europa* selected. Remember to save the changes if you change printer! You can assign more printers to this location by clicking **Add printer** again to bring up another printer line. To remove a printer you select the **Delete** checkbox corresponding to the printer(s) you want to delete, and click **Save** to apply the changes. The printer(s) will disappear.

**Note:** Printers contains entries for all available printers. These entries are just shadows of the real CUPS (Common Unix Printing System) printer system entries. This means that you first need the printers to be installed in the CUPS printer system and then added to this list.



## 16.3.6. The Desktop Customizer Module

The ThinLinc Desktop Customizer is described more fully in its own chapter, Chapter 17. Links to sections of this chapter pertaining to the respective menus of the Desktop Customizer Module are provided below for convenience.

### 16.3.6.1. Application Groups

For information on configuring Application Groups using TLDC, see Section 17.2.5

### 16.3.6.2. Applications (Manual)

For information on configuring Manual Applications using TLDC, see Section 17.2.3

### 16.3.6.3. Applications (System)

For information on configuring System Applications using TLDC, see Section 17.2.1.1

### 16.3.6.4. Menu Structure

For information on configuring Menu Structures using TLDC, see Section 17.2.4

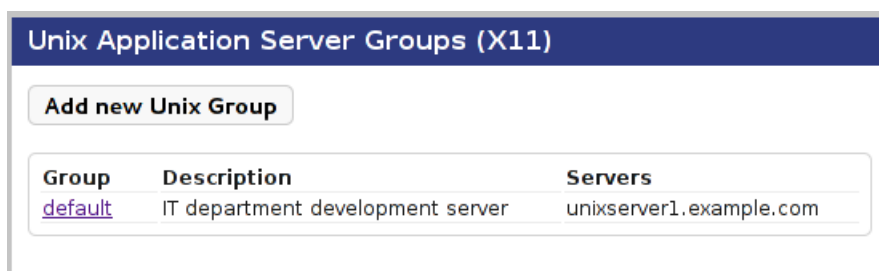
## 16.3.7. The Application Servers Module

The Application Servers module provides a way to define application server groups via `tlwebadm`. For more information on application server groups, see Section 14.2.4.

### 16.3.7.1. UNIX Application Server Groups

This menu allows you to configure UNIX-based application server groups, for publishing applications via X11-forwarding. The default installation provides a single example server group, as shown in Figure 16-5.

**Figure 16-5. UNIX Application Server Groups List**



Unix Application Server Groups (X11)		
Add new Unix Group		
Group	Description	Servers
<a href="#">default</a>	IT department development server	unixserver1.example.com

To edit an existing server group, click on the name of the group in the list. To add a new server group, click on the **Add new Unix Group** button - Figure 16-6 shows the page for adding new UNIX server groups.

**Figure 16-6. Adding a UNIX Application Server Group**

**Unix Application Server Groups (X11)**

**Add new Unix Group**

Group	Description	Servers
<b>Group name</b>	New Unix Group	
<b>Description</b>		
<b>Servers (separate with space)</b>		
<input type="button" value="Delete"/> <input type="button" value="Save"/>		
<input type="checkbox"/> Yes, really delete New Unix Group		
<a href="#">default</a>	IT department development server	unixserver1.example.com

The descriptions of the fields are provided in Table 16-2 below.

**Table 16-2. UNIX Application Server Group Fields**

Name	Description
Group name	A unique name for the application server group.
Description	A description of the application server group.
Servers	A space-separated list of servers (IP addresses or resolvable hostnames) which will be members of this application server group.

### 16.3.7.2. Windows Application Server Groups

This menu allows you to configure Windows-based application server groups, for publishing Windows applications and desktops via RDP.

To edit an existing server group, click on the name of the group in the list. To add a new server group, click on the Add new WTS Group button - Figure 16-7 shows the page for adding new Windows server groups.

**Figure 16-7. Adding a Windows Application Server Group**

The screenshot shows a web interface titled "Windows Terminal Server Groups (RDP)". Below the title is a button labeled "Add new WTS Group". The main form is divided into two columns: "Group" and "Servers". The "Group" column contains the following fields: "Group name" (with the value "New WTS Group"), "Description", "Servers (separate with space)", "Domain", "Keyboard layout" (with the value "sv"), "Sound system" (with radio buttons for "auto", "padsp", "esddsp", and "disabled", where "auto" is selected), "Redirect printers via RDP" (with a checked checkbox and the label "Yes"), "Use Novell username even when reconnecting" (with a checked checkbox and the label "Yes"), and "Extra rdesktop arguments". The "Servers" column is currently empty. At the bottom of the form are two buttons: "Delete" and "Save". Below the "Delete" button is a checkbox labeled "Yes, really delete New WTS Group".

The descriptions of the fields are provided in Table 16-3 below.

**Table 16-3. WTS Application Server Group Fields**

Name	Description
Group name	A unique name for the application server group.
Description	A description of the application server group.

Name	Description
Servers	A space-separated list of servers (IP addresses or resolvable hostnames) which will be members of this application server group.
Domain	The Windows domain, if any, to specify when connecting.
Keyboard layout	The keyboard layout to use for this server group.
Sound system	The sound system to use for this server group, or select "disable" to disable sound.
Redirect printers via RDP	Check this box to share printers via RDP from the ThinLinc server.
Use Novell username even when reconnecting	Check this box to reconnect using the Novell username by default.
Extra rdesktop arguments	Any extra arguments you wish to give to rdesktop can be entered here.

### 16.3.8. The Novell Configurator Module

The ThinLinc Novell Configurator is described more fully in its own chapter, Section 9.3.

## Chapter 17. Building Custom Linux Desktops with the ThinLinc Desktop Customizer

In this chapter, we will document how to create custom desktops for ThinLinc users using either the K Desktop Environment (<http://www.kde.org>) or the Gnome Desktop Environment (<http://gnome.org>), in combination with the ThinLinc Desktop Customizer (TLDC).

The TLDC's core functionality is to build the menu of ThinLinc users based on factors such as group membership, user name and ThinLinc profile. It can also add icons to the desktop of each user, based on the same premises.

### 17.1. Introduction

The ThinLinc Desktop Customizer is a combination of a web-based administration tool and a command that is run at session startup for all users. It enables the administrator to decide what menu entries should be presented for specific users, and what icons should be made available on the desktop. Which menu entries and/or desktop entries are given to a specific user is decided based on the unix group memberships of the user, the username and what ThinLinc profile was chosen (if any).

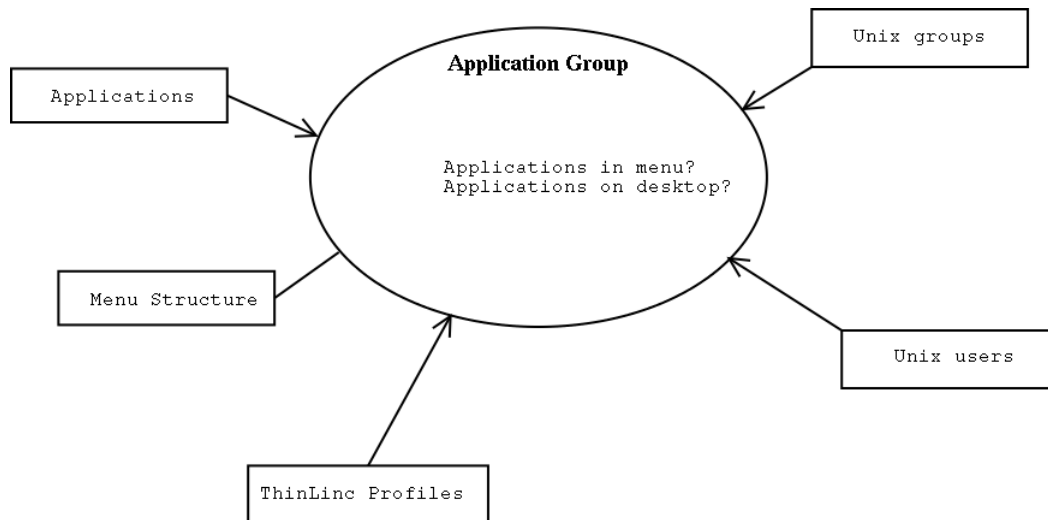
**Note:** Since KDE4 uses a different desktop configuration mechanism to previous versions of KDE, icons added to the desktop using the ThinLinc Desktop Customizer will not be shown in KDE4. This can be solved by changing the "Desktop layout" setting of your KDE4 desktop to "Folder view".

### 17.2. Using the ThinLinc Desktop Customizer

Using the ThinLinc Desktop Customizer, the system administrator can decide what applications should be available in the menu and/or on the desktop for specific users or for users that are members of some Unix group. The ThinLinc Desktop Customizer is configured via a web interface, part of the ThinLinc Web Administration. Chapter 16 describes the interface in general, this section will describe the Desktop Customizer part of it.

### 17.2.1. Concepts

Figure 17-1. ThinLinc Desktop Customizer Concepts



The main concept in the ThinLinc Desktop Customizer is the *Application Group*. The Application Group combines data about Applications, A Menu Structure, Unix Groups and Users, and some other Settings.

#### 17.2.1.1. Applications

The Applications referred to in the Application Groups are found by scanning the directories defined in the Hiveconf parameter `/utils/tl-desktop-customizer/xdg_data_dirs` for files named `*.desktop`. The files are read according to the Freedesktop.org (<http://freedesktop.org/>) Desktop Menu Specification. The TLDC scans the directories in the same way that KDE will do when building the menu.

Some applications are marked by the system to be shown only for root, or only in either Gnome or KDE. On SuSE, there is also a `X-SuSE-Unimportant` parameter in some `*.desktop` files, which will make the KDE packaged with SUSE hide the application. TLDC handles this by adding a comment to the application in the applications listing, and in the selectboxes used when creating application groups.

In addition to the desktop files automatically found, it is also possible to manually define applications. This is needed for example when an application without a `*.desktop`-file has been installed, when an application has been installed in a non-standard location, or when the applications are available on application servers and should be run for example via **tl-run-winapp**.

### 17.2.1.2. Menu Structure

Each Application Group can add applications to a specific place in the menu structure. The available menu structure is edited in the "Menu structure" part of the web based administration interface.

### 17.2.1.3. Unix Groups and Users

An Application Group is used by zero or more Unix groups and by zero or more specific users. An example would be an educational environment. Let's say that all pupils attending the class "biology 4" are members of the Unix Group "bio4". By creating an Application Group named "Biology 4" with all Applications that are specific to the biology class, and then adding the "bio4" Unix Group as one of the groups that should be assigned the "Biology 4" Application Group, all students attending the class will automatically get the applications specific to the biology class in their menu. By adding the teacher of the class as a specific user, he/she as well will also get access to the applications.

## 17.2.2. Using the ThinLinc Desktop Customizer

The Daily use of the TLDC consists of one or several of the following steps:

- Create an Application
- Create a folder in the Menu Structure
- Bind one or several applications to a folder in the menu structure, using an Application Group

In the following sections, we will more thoroughly describe the different actions that may be needed.

### 17.2.3. Handling Applications

The handling of applications is normally the first step in using the TLDC. Click on the "Applications (Manual)" link in the TLDC, and you will enter a view where the applications you've defined manually are listed. Several example applications are included with ThinLinc at installation. By clicking on the text "Applications defined by system", you can also see what applications are found automatically by scanning, as described in Section 17.2.1.1.

If the application you want to add to a menu or to the desktop is not found among "Applications defined by system", you need to define it manually. This is the case for applications installed without adding a .desktop file in the correct location, or for applications that are run by connecting to an application server, for example a Windows Terminal Server.

Defining applications manually is done by clicking on the button "Add new application" (located at the top of the list of applications). This leads to a page where you can define the following properties for the new application:

- *Default Application Name*

This is the name of the application. It's the name that is written next to the icon (if any), in the menu, and under the icon if the application is to be added to the desktop.

The Default Application Name is used if there is no name defined for the language in use when the application is shown, or if the language is english.

- *Application Name (<language-code>)*

This is the name of the application in the language with the RFC1766 language code <language-code>. This name is shown if the locale is set to that language when the menu or desktop is shown.

The languages that should be configurable are set by editing the parameter `/utils/tl-desktop-customizer/desktop_languages`. The default value of this parameter is `sv`, which means that the TLDC will allow you to set the default name and the name in swedish.

- *Command*

The Command field is divided into three sections. Which one to choose depends on which kind of application you are adding.

- *Command Run on ThinLinc Server*

Select this radio button if you are adding an application that is executed on the ThinLinc server. Enter the path to the command followed by any arguments in the *Command* box. The input box follows bourne shell syntax rules.

Example:

```
"/usr/bin/my program" --fullscreen --title "My title"
```

- *Command Run on Windows Terminal Server in SeamlessRDP mode*

Select this radio button if you are adding an application that is run on a Windows Terminal Server in SeamlessRDP mode. In SeamlessRDP mode, an application running on a WTS can open several windows on the ThinLinc desktop, and windows can be resized dynamically. Applications defined this way are run via the command **tl-run-winapp-seamless**.

Enter the path to the command on the Terminal Server's filesystem in the *Command on Server* input box. If the path contains spaces, quote the entire path with double quotes.

Example:

```
"C:\Program Files\My Vendor\My Program.exe" %TEMP%
```

Choose which application server group should be used to run the command in the selectbox labeled *Application Server Group*. By having several application server groups, you can have several groups of Windows Terminal Servers for different purposes, with individual load balancing between the servers in each group.

Add any extra arguments to **tl-run-winapp-seamless** in the third input box, labeled *Other tl-run-winapp-seamless args*.

- *Command run on Windows Terminal Server in Standard Mode*

Select this radio button if you are adding an application that is run on a Windows Terminal Server (from now on referred to as *WTS*) in standard mode (not in SeamlessRDP mode). Applications defined this way are run via the command **tl-run-winapp**.

Enter the path to the command on the Terminal Server's filesystem in the *Command on Server* input box. If the path contains spaces, quote the entire path with double quotes.

Example:

```
"C:\Program Files\My Vendor\My Program.exe" %TEMP%
```



Choose in which application server group the command should be run in the select box labeled *Application Server Group*. By having several application server groups, you can have several groups of Windows Terminal Servers for different purposes, with individual load balancing between the servers in each group.

It's most often recommended to hide window manager decorations, or the application will get double frames.

When running applications on WTS in this mode, the application gets a fixed size. The size can be set to one of *fullscreen*, *size of desktop work area* (which is fullscreen minus the area used by the window manager's taskbar), a fixed size or a percentage of the ThinLinc desktop's size.

The title of the window is by default set to the name of the executable run on the WTS. The title can be explicitly set in the *Window Title* input box.

Add any arguments to **tl-run-winapp** in the last input box, labeled *extra tl-run-winapp args*.

- *Path to Icon file*

The filename of the Icon for the application. If the icon is available in one of the directories where KDE automatically looks for icons, just the filename without the extension can be given. Otherwise, the complete path must be specified.

- *Command Startup Feedback*

Check the box to instruct the Window Manager to show a special icon while the command is starting. Note that this does not work very well for applications run from a Windows Terminal Server, so for these applications, it's recommended not to enable this functionality.

Press save when done filling the fields. The application will now show up among the other manually defined applications.

If you want to, you can add the application directly to an existing application group by checking the checkbox in front of the application name, then selecting the application group and if the application should be added to the menu or desktop of this application group, in the form at the top of the page. This can be done for both manually defined, and automatically found applications.

## 17.2.4. Defining a Menu Structure

With TLDC, the normal menu structure as defined by the Linux Distributor or by the KDE team is not used. Instead, a new menu structure is defined. This gives more flexibility in designing menus. The TLDC administrator can fully decide where in the menu structure a certain application is placed.

To define the menu structure, click on the "Menu structure" submenu in the left pane of the TLDC administration interface. This leads to a view where a menu structure can be defined. The Root menu folder is always available and can't be removed.

**Note:** A menu called "*Hidden Menu*" is shipped with the default ThinLinc configuration. See Section 17.4.3 for an explanation of its functionality. Please don't remove it if you are planning to use KDE.

The following properties can be edited for a menu:

- *Default Menu Name*

This is the name of the menu, as it will be shown in the menu.

- *Menu Name (<language-code>)*

This is the name of the menu in the language with the RFC1766 language code <language-code>. This name is shown if the locale is set to the language at runtime.

- *Path to Icon File*

The filename of the Icon for the menu, shown to the left of the menu name in the KDE menu. If the icon is available in one of the directories where KDE automatically looks for icons, just the filename without the extension can be given. Otherwise, the complete path must be specified.

- *Hide This Menu*

If this radio button is set to *Yes*, the menu will be a hidden menu. It will not be shown in the menu, but any applications that are added to this menu via an application group will be available in the KDE File Associations.

Just as for Applications, the name of the menu can be defined in several languages. The *Default Menu Name* is used if no language-specific name is defined, or if the locale specifies that the language is english. The list of languages that can be defined using the TLDC is found in `/utils/tl-desktop-customizer/desktop_languages`.

### 17.2.5. Defining Application Groups

Enter the "Applications Groups" part of the "Desktop Customizer". This will present you with a list of existing application groups and their settings.

**Note:** An application group called "*Hidden*" is shipped with the default ThinLinc configuration. See Section 17.4.3 for an explanation of its functionality. Please don't remove it if you are planning to use KDE.

Press the button "Add new group" (located at the top in the table of existing application groups) to create a new application group. This will open a rather large form, where you can define the following properties:

- *Name of the Application Group*

This is the name of the Application Group. This is not displayed to the users, but only to the System Administrator using the ThinLinc Desktop Customizer. Set to something that reflects the contents of the Application Group.

- *Applications Added to Menu*

First, define in the dropdown box what location in the menu structure applications chosen in the boxes below it should be added to.

Add to the left selectbox the applications that should appear in the menu folder selected above, for the users that are assigned this Application Group. The right selectbox lists the applications defined or found installed on the system. If there are no applications available, you've forgotten to define Applications, as documented in Section 17.2.5.

- *Applications Added to Desktop*

Add to the left selectbox the applications that should appear as icons on the desktop of the users that are assigned this Application Group. Just as for Application added to the menu, only applications earlier defined, or automatically found, will show up as selectable.

- *Unix Groups with this Application Group*

This is where you connect Unix groups to Application Groups. If for example a specific school should be assigned this Application Group, and all the pupils of that school are members of the Unix group "school-1", add the Unix group "school-1" to the left selectbox. When logging in, the group memberships of each user is inspected to determine which Application Groups to assign to the user.

**Note:** If the mapping between the numerical group id and the group name doesn't work, the group is shown as `#<gid>`. This might be because the group has been removed from the system, or because the operating system has problems in the connection to the directory service used.

- *Specific Users with this Application Group*

This parameter allows you to decide that specific users should be assigned this Application Group as well, even if they are not a member of one of the groups that were added above. This way, for very specialized applications, no Unix group needs to be created. Another way of using this field would be that the teachers of a specific class could be added to the Application Group for that class, if the teachers are not part of the unix group that is associated with the class.

**Note:** If the mapping between the numerical user id and the user name doesn't work, the user is shown as `#<uid>`. This might be because the user has been removed from the system, or because the operating system has problems in the connection to the directory service used.

- *ThinLinc profiles with this Application Group*

This setting allows you to connect the Application Group to ThinLinc Profiles as documented in Section 14.4.3. This allows for different Application Groups to be selected based on user input after login.

- *Shell Command Activating this Application Group*

This setting allows you to activate application groups based on the return value of an arbitrary command. If the command returns 0 (which is the standard return code for success for shell commands), the application group will be activated.

This can be used for example to activate application groups based on OU in eDirectory installations by using the **tl-nds-memberof-container** command. It can also be used to activate an application group for all users by running **/bin/true** as activation command.

The command is run via the shell in the current user's environment when running **tl-desktop-activate.sh**. The environment variable **TLDCGROUP** is set to the application group currently under consideration for activation.

- *Save!*

Don't forget to press the Save-button, or none of the changes will be written to the database.

### 17.2.6. Distribute Configuration to all agent hosts

After doing changes to the Desktop Configuration, the new configuration must be copied to all VSM agent hosts. The files/directories to be copied are

`/opt/thinlinc/etc/conf.d/tl-desktop-customizer.hconf` and all subdirectories of `/opt/thinlinc/desktops`.

**Best Practice:** Use the **tl-rsync-all** command as described in Chapter 13 to copy the files.

## 17.3. Enabling the Custom Desktops for users

Enabling the Custom Desktops for users is easy. Simply create a symbolic link in `/opt/thinlinc/etc/xstartup.d`:

```
# ln -s
    /opt/thinlinc/bin/tl-desktop-activate.sh
    /opt/thinlinc/etc/xstartup.d/35-tl-desktop-activate.sh
```

The ThinLinc session startup will read this file and make sure the environment variable **KDEDIRS** is set correctly. It will also write a `~/.config/menu/applications.menu` and possibly create symbolic links under `~/.kde/share/apps/kdesktop/Desktop`. Your profile should then execute the command **startkde**.

**Note:** The TLDC activation script only runs TLDC for non-root users. Test your TLDC configuration using a normal user.

## 17.4. Tips & Tricks with TLDC

### 17.4.1. Unwanted Icons on the Desktop with KDE

At first login for each user, KDE copies files from `/usr/share/apps/kdesktop/DesktopLinks` to the Desktop directory of the user. This means that if there is a Home Icon in `DesktopLinks` and you add a Home Icon via TLDC, there will be two Home Icons.

Remove the contents of `/usr/share/apps/kdesktop/DesktopLinks` to solve the problem, and let TLDC be the sole provider of icons on the desktop.

**Note:** If you KDE is based somewhere else than under `/usr`, the `DesktopLinks` directory will be situated elsewhere. For example, on SuSE, KDE is based at `/opt/kde3`.

### 17.4.2. File Associations for Applications Not In the Menu

When KDE tries to determine what application to use for opening a specific file, it is only looking for applications that are available in the menu. There are cases where not all applications that may be used for opening files are meant to be available in the menu.

In this case, create a hidden menu by setting *"Hide this Menu"* to *Yes* in the Menu Structure Editor, and then create an Application Group that adds the applications that should be available for file associations in to this menu.

### 17.4.3. Home Icon not Working in KDE?

This is a case of the problem above where File Associations are not working. Create an Application Group that includes the Konqueror (`kde-kfmclient_dir`) application in a hidden menu, and make sure this application group is added for all relevant users, and the home icon will work again.

**Note:** A menu named *"Hidden Menu"* is created by the application group *"Hidden"* which is by default activated for the profile *kde*. This menu contains the `kde-kcmclient_dir` to make sure the home icon is working. Make sure this application group is activated for all users with a desktop based on KDE.



## Appendix A. TCP Ports Used by ThinLinc

### A.1. On Machine Running VSM Server

#### 22: SSH Daemon

Port 22 is not used by ThinLinc *per se*, but since no ThinLinc installation can work without a running SSH daemon, we list port 22 here. Port 22 is the normal SSH port, but basically any port can be used - the client has support for connecting to any port. Note however that if the SSH daemon on the VSM server is listening on port *x*, all VSM agents must also have their SSH daemons configured to listen on port *x*.

#### 300: ThinLinc HTML5 Browser Client

By default, ThinLinc's HTML5 Browser client service `tlwebaccess` is available on TCP port 300. Traffic to this port is encrypted (TLS).

**Note:** The port on which `tlwebaccess` runs is configurable via the parameter `/webaccess/listen_port`. See Section 8.7.3.1.1 for details.

#### 1010: ThinLinc Administration Interface (`tlwebadm`)

By default, ThinLinc's web-based administration interface is available on TCP port 1010. In order to access this interface remotely, port 1010 will need to be reachable. Traffic to this port is encrypted (TLS).

**Note:** The port on which `tlwebadm` runs is configurable via the parameter `/tlwebadm/listen_port`. See Section 16.2 for details.

#### 9000: VSM server

The VSM server listens on port 9000. The traffic is not encrypted, but sensitive information will only be shared with root or connections originating from a port lower than 1024, from a list of known IP addresses. The protocol used is XML-RPC through HTTP (using a minimal internal HTTP server in the VSM server).

### A.2. On Machine Running VSM Agent

#### 22: SSH Daemon

Just as for the VSM server, there must be a SSH Daemon running on all VSM agent machines. This daemon is normally listening to port 22, although it can listen to other ports as well. See the entry about port 22 on Section A.1.

300: ThinLinc HTML5 Browser Client

By default, ThinLinc's HTML5 Browser client service `tlwebaccess` is available on TCP port 300. Traffic to this port is encrypted (TLS).

**Note:** The port on which `tlwebaccess` runs is configurable via the parameter `/webaccess/listen_port`. See Section 8.7.3.1.1 for details.

904: VSM Agent

The VSM agent listens on port 904 for incoming requests from the VSM server host. The traffic is not encrypted, but the VSM agent only allows connections originating from a port lower than 1024, from a list of known IP addresses. The protocol in use is XMLRPC through HTTP.

1010: ThinLinc Administration Interface (`tlwebadm`)

By default, ThinLinc's web-based administration interface is available on TCP port 1010. See the entry about port 1010 at Section A.1.

5901-5999: VNC servers for first 99 sessions

Ports 5901-5999 are used by `Xvnc` processes serving display numbers strictly below 100.

4900-5899: Tunnels from Terminal Server to clients

The ports in this interval is used as serverside endpoints for the SSH tunnels used to access local resources of the client, for example local drives, serial ports and sound.

This interval is used for sessions with display number strictly below 100.

The algorithm used for calculating which ports to use for a specific display number in this interval is  $4900 + display-id * 10 + SERVICE\_SLOT$  where `SERVICE_SLOT` is a number defined under `/vsm/tunnelservices`.

6001-8000: X Display ports

If `Xvnc` is configured to listen for incoming TCP requests from X Window System clients on other hosts, ports 6001-8000 are used by display numbers 1-2000. The default is not to listen for incoming TCP requests, so in the default configuration, the ports in this interval are not open.

Port 32767 downwards to 11857

The algorithm described below is used to allocate ports for the `Xvnc` process and for the serverside endpoints for SSH tunnels to access local resources of the client. This algorithm is used for sessions with display numbers strictly higher than 99.

Each session is allocated `/vsm/tunnelslots_per_session` (default value 10) + 1 ports, leading to an allocation of 11 ports per session with the default configuration. The ports are allocated starting with the port given as `/vsmagent/max_session_port` (default 32767), and then downwards. This means that the ports are aligned upwards as closely as possible to the upper limit defined. This is a good practice to avoid collisions with other services running on the machine.

Some examples follow



Display number 50

The VNC port will be 5950 which is  $5900 + display$ .

The tunnel ports allocated for this display are 5400-5409, which is  $4900 + (10 * display) + SERVICE\_SLOT$  where `SERVICE_SLOT` is 0-9.

Display number 100, `/vsmagent/display_min = 1` (the default),  
`/vsmagent/max_session_port = 32767`.

The VNC port will be 32757, which is  $32767 - ((display - 100) * (/vsm/tunnelslots\_per\_session + 1) + /vsm/tunnelslots\_per\_session)$ .

Ports 32758-32767 (inclusive) will be used for tunnel ports.

Display number 300, `/vsmagent/display_min = 100`, `/vsmagent/max_session_port = 32767` (the default).

The VNC port will be 30557 which is  $32767 - ((display - 100) * (/vsm/tunnelslots\_per\_session + 1) + /vsm/tunnelslots\_per\_session)$ .

Ports 30558-30567 (inclusive) will be used for tunnel ports.

Display number 600, `/vsmagent/display_min = 300`, `/vsmagent/max_session_port = 32767` (the default).

The VNC port will be 29457, which is  $32767 - ((display - 300) * (/vsm/tunnelslots\_per\_session + 1) + /vsm/tunnelslots\_per\_session)$ .

Ports 29458-29467 (inclusive) will be used for tunnel ports.

### A.3. On Windows Terminal Servers

5667: ThinLinc Load Agent

The ThinLinc Load Agent for Windows Terminal Servers listens on port 5667, and reports system load to the ThinLinc servers.



## Appendix B. Troubleshooting ThinLinc

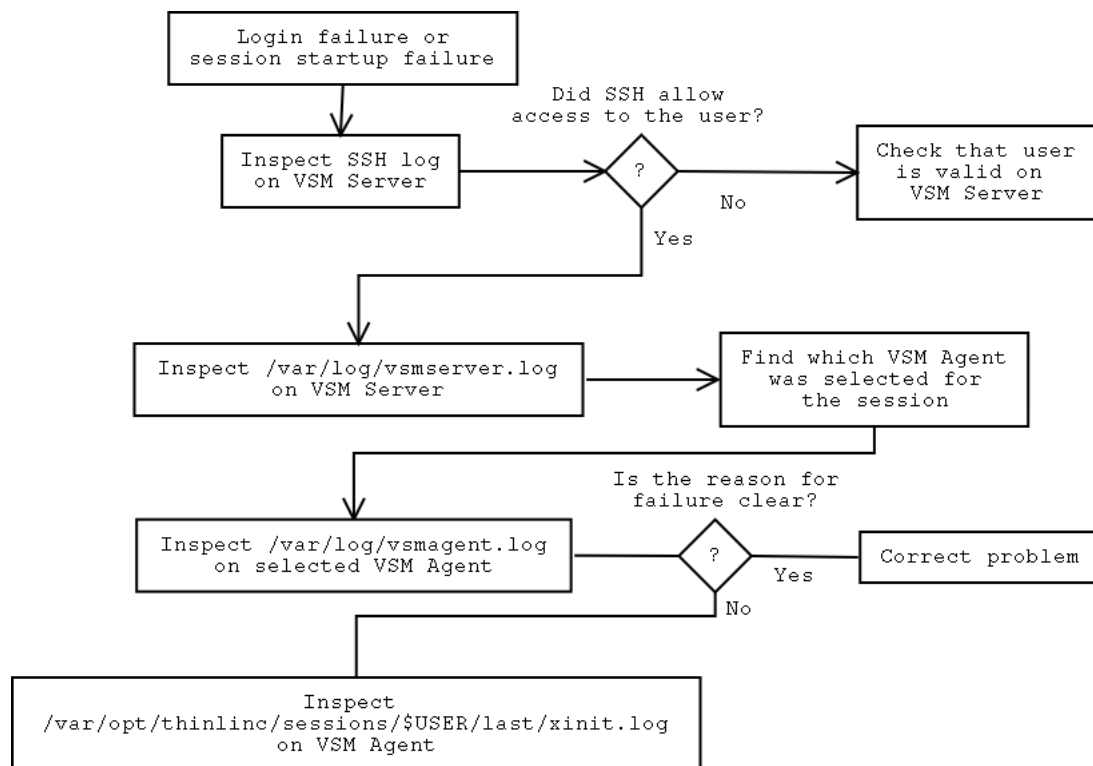
In this appendix, we will describe how to troubleshoot common problems in a ThinLinc installation.

We will begin by giving a general view of the recommended troubleshooting method, and then continue with more detailed instructions for troubleshooting specific problems.

### B.1. General troubleshooting method

In most cases, troubleshooting a ThinLinc session problem should follow the method outlined in Figure B-1.

**Figure B-1. The General Troubleshooting Method**



The method is to first check that the user was let in by SSH on the VSM server. This information is found on different places on different distributions. Common log filenames for SSH information are `/var/log/secure`, `/var/log/auth.log` or `/var/log/daemon.log`. If the user was let in by SSH, the VSM server log (`/var/log/vsmserver.log`) is inspected. In some cases, the reason for session failure can be found there, but most of the times, it's necessary to find out which VSM agent was selected for the session, and inspect the VSM agent log (`/var/log/vsmagent.log`) on the server in question.

If inspecting `/var/log/vsmagent.log` on the server that was selected for the session does not reveal the reason for the failure, there is a per-session log in `/var/opt/thinlinc/sessions/<username>/last/xinit.log` where the output of commands run during session startup is stored.

In very rare cases, it might also be necessary to inspect the SSH log on the VSM agent.

## B.2. Troubleshooting Specific Problems

### B.2.1. Problems Where the Client Reports an Error

In the following sections, we will describe how to cope with problems where the ThinLinc client is reporting an error.

#### B.2.1.1. Couldn't set up secure tunnel to ThinLinc server. (Couldn't establish SSH tunnel, SSH terminated.)

This error is caused by failure to connect to the SSH daemon on the ThinLinc server (the server running the VSM server). This could be caused by the fact that the SSH daemon is simply not running, or that it is not letting the user in for some reason.

Another possible reason is that there is a firewall between the user and the ThinLinc server, that forbids communication.

#### B.2.1.2. "Login Failed! Wrong username or password."

This error is very often caused simply by the user entering an incorrect password. Begin by verifying that the user is actually entering the correct username and password.

If the username and password are correct and this is the first time the user tries to login, check the SSH logs of the server. If SSH says that the user is invalid, that means that something is incorrect in the user's user information database entry. For example, this may happen if a user stored in eDirectory has two *cn* attributes, one of them different than the other.

The **getent** command can be a valuable tool to dissect problems of this type. If the output of **getent passwd <username>** doesn't produce any output, that is a sign that the user is in fact invalid.

**Note:** Usernames beginning with numbers (for example *96aabbcc*) are parsed as numeric uids by **getent**, rendering **getent** rather useless for debugging purposes in environments with username schemes beginning with numbers.

If usernames with numbers in the beginning are in use, the following python code can be used to verify a username

```
python-thinlinc -c 'import pwd, sys; print pwd.getpwnam(sys.argv[1])' <username>
```

### **B.2.1.3. The SSH connection succeeded, but the ThinLinc server connection failed. Perhaps this server doesn't run a ThinLinc server?**

This error is most often caused by the fact that the VSM server is not running on the server. Start the VSM server and try again.

A user entering the wrong hostname, for example the hostname of one of the VSM agents, would also get this error message. Check that the user has entered the correct hostname. In very rare cases, this could also be caused by incorrect DNS data.

### **B.2.1.4. ThinLinc login failed. (Couldn't create session on selected server)**

Check the VSM server log to see which VSM agent was selected. Check the VSM agent log on the selected server to see what happened.

One common reason for this error is that the user database on the selected server is failing, so the user does exist on the VSM server, but not on the VSM agent.

### **B.2.1.5. ThinLinc login failed. (Cannot find any working terminal server).**

This error is reported if there were no working VSM agents available according to the load balance information in the VSM server.

In a system with few VSM agent servers, restoring a VSM agent that has been down for some reason doesn't take effect immediately - the load balance information is only updated once every 40 seconds by default. Either wait for the load balance cycle to complete, or restart the VSM server. In a small cluster it might be a good idea to lower the load balance cycle, by setting the parameter

`/vsmserver/load_balance_cycle.`

The load balance information can be inspected in the ThinLinc Web Administration, see Chapter 16.

### **B.2.1.6. ThinLinc login failed (The terminal server couldn't create your session)**

When this error occurs, the user was valid on the VSM server, but for some reason, the session couldn't be created on the VSM agent.

One very common reason for this problem is that the VSM agent has lost its connection to the user database backend (LDAP, Windows domain or other database), so the user exists on the VSM server, but not on the VSM agent. If this is the case, the VSM agent log on the selected server will clearly state that the user doesn't exist on the system.

Another very common reason is home directory trouble on the VSM agent. Verify that the home directory exists on the selected server, and that it is owned by the correct uidNumber/gidNumber. Of course, the user must have write permissions on his/her home directory.

To verify that the home directory works, the following command can be used:

```
ssh <username>@<agenthost> touch .
```

If the home directory is correctly mounted and writable by the user, the above command will not produce any output except the password question.

#### **B.2.1.7. Couldn't set up secure tunnel to VNC! (Couldn't establish SSH tunnel, SSH terminated.)**

This error is caused by failure to connect to the SSH daemon on the selected VSM agent. This could be caused by the fact that the SSH daemon is simply not running, or that it is not letting the user in for some reason.

Another possible reason is that there is a firewall between the user and the selected VSM agent that disallows communication.

### **B.2.2. Problems that Occur After Session Start**

In this section we will discuss some problems that can occur after the successful login, that is, after the ThinLinc login window has closed. In this phase, a number of session startup problems can occur

#### **B.2.2.1. Session starts, but closes down immediately**

If the ThinLinc login window closes, and the session starts up but then immediately shuts down, inspect `xinit.log` found in `/var/opt/thinlinc/sessions/<username>/last/` on the selected VSM agent. Some of the commands run during session startup will probably have written an error message that will be stored in that file.

It may also be of value to inspect the VSM agent log on the selected server.

#### **B.2.2.2. At login, user is reconnected to previous, faulty, session**

If a previous session still exists and is faulty, for example because of desktop environment failures, the user is reconnected to the same session when logging in. Disconnect from the session, enable the *"End existing session"* checkbox and log in again. That will terminate the current session and start a new one.

#### **B.2.2.3. Login Succeeds, but the ThinLinc Desktop Configuration fails**

When using the ThinLinc Desktop Customizer, as documented in Chapter 17, the KDE or Gnome menu and the entries on the desktop are customized at each login. If this fails, quota problems are very often the problem. Check the quota of the user in question.

#### **B.2.2.4. Login Succeeds, but KDE Fails to Start**

If KDE fails to start, complaining about being unable to create symlinks and similar, quota problems are very often the real problem. Check the quota of the user in question.

## Appendix C. Manually Configuring Integration with Novell eDirectory

In Section 9.3, we explain how to integrate a ThinLinc Cluster with Novell eDirectory using the ThinLinc Novell Configurator (TLNC). In this chapter we will explain how to do it manually, without the TLNC.

This information is provided to further explain what is done by the TLNC, and to provide documentation on what needs to be done if the TLNC can't be used for some reason.

### C.1. Schema extensions

In order for *pam\_ldap* and *nss\_ldap* to work, the NDS must have schema extensions for *posixAccount* and *posixGroup*.

Most eDirectory installations already have the relevant schema installed, but if not, it must be installed.

The extensions will be added to NDS if Novell Native File Access for Unix is installed. During installation of this the command **schinst** is run to extend the schemas with *posixAccount* and *posixGroup*. You may have to run **schinst** manually later, for example after applying a Novell service pack. If Novell Native File Access for Unix is not available for your Novell platform (it's available for Netware 6.0 and above), extending the schema by hand should work although this is an untested configuration.

### C.2. Increasing performance by adding an index on some Attributes.

In a standard eDirectory installation, there is no index on the Object Class attribute, nor on the *uidNumber* or *gidNumber* attributes. Since *pam\_ldap* and *nss\_ldap* use LDAP queries that include the *objectclass*, *uidNumber* and *gidNumber* attributes, performance can often be increased by adding an index for these specific attributes. This has to be done on all servers that carry data needed for authentication.

We will give an example on how to add an index on "Object Class":

1. Find the server object for the server where the index is to be added. That is, find the Netware server object, not the LDAP server object.
2. Select properties, and choose the index tab.
3. Press the Add button, set Index Name to "objectclass". Choose "Object Class" in the list of Attributes.

**Note:** The list of Attributes has a peculiar sorting order. All attributes that have names that start with a capital are ordered before the ones that don't.

Set the Rule to "Value".

4. Press OK.
5. Press Apply. That will create the index, a background process that may take a few minutes on a very large directory.

Repeat the procedures above for all servers that have replicas with the relevant information.

Repeat the steps above for the *uidNumber* and *gidNumber* attributes, adjusting parameters as needed.

### C.3. Removing Attribute Mappings

After extending the schema, two attribute mappings must be removed for proper operations. If they are not removed, neither *uidNumber* nor *gidNumber* will work as needed.

Attribute mappings are used to map LDAP attribute names to NDS attribute names as a compatibility feature of Novell NDS. Since NDS has been around for a longer time than the LDAP specification, a lot of software exists that use the NDS names of object classes and attributes.

In order for LDAP authentication to work, the mapping from *uidNumber* to *UID* as well as the one from *gidNumber* to *GID* must be removed.

To remove the mappings you right click on the LDAP group entry at root level of your tree and select **Properties**. Under the **Attribute map** tab is a list of attribute mappings. Depending on your version of eDirectory and your version of Console One, there are either direct mappings between *gidNumber* and *GID*, or a mapping between *groupID* and *GID*, with *gidNumber* as a secondary LDAP attribute. If there is a search function available, use that to locate the relevant mappings, and delete them.

One symptom of the fact that attribute mappings for *gidNumber* and *uidNumber* have not been removed is that when searching eDirectory for groups without specifying what attribute to fetch, the *gidNumber* shows up, but when explicitly specifying that *gidNumber* should be fetched, no data is returned.

### C.4. Adding `nss_map_attribute` statements to `/etc/ldap.conf`

For NSS (looking up information about users and groups) to work well when fetching information from eDirectory, the following three lines should be in `/etc/ldap.conf`

```
nss_map_attribute uniqueMember member
nss_map_attribute uid cn
pam_password nds
```

The first two make sure that the Linux machine asks for the correct attributes, and more importantly, that when processing a request to find out which groups a user is member of, it doesn't have to lookup every DN found as a member to find out which uid it corresponds to. The last line makes sure changing passwords in eDirectory from Linux works. Use any PAM-enabled password-changing program in Linux to achieve this functionality.

### C.5. Creating a DN for search operations

In most environments, it's not a good idea to setup eDirectory so that anyone can read the attributes needed by LDAP Authentication (*uid*, *uidNumber*, *gidNumber*, *homeDirectory* and *loginShell*). Depending on the network setup, the information may be more or less sensitive. To prevent this, a special user is created in the database, and all search operations from the ThinLinc servers are made after binding as this user. This way, the amount of information that can be extracted by an anonymous user is limited. However, all users on the ThinLinc servers can read the password of this user, so the protection is limited. The user must have access to the mentioned attributes. In this section, we will describe how to create this user and setup the access control.



- Begin by creating a user in eDirectory at an appropriate place in the tree. This user is a so-called *application DN*, so if your tree already has a place for similar users, place the user there. In our example, the DN of the user will be `cn=tl-posixsearcher,ou=thinlinc,o=example`. This DN will be used as `binddn` in `/etc/ldap.conf` on the ThinLinc server.
- Set a password for the user using the normal procedure (choose properties on the object, set the password under the Restrictions tab). Do not use a password that is used for anything else. This password will be used as `bindpw` in `/etc/ldap.conf` on the ThinLinc server and will be readable by all users on the ThinLinc server.
- Now add the user just created as a trustee on a appropriate object in the tree. This object should be above all users that should be able to login to the ThinLinc server(s). For example, if all users reside under `ou=People,o=example`, add the user as a trustee on the `ou=People,o=example` object. Adding the trustee is done using the following procedure:
  - Right-click the object where the trustee should be added. Select "Trustees of this object..."
  - Press "Add Trustee..." in the dialog that appears and select the user we just created (`cn=tl-posixsearcher,ou=thinlinc,o=example` in our example).
  - Select the user just added and press "Assigned Rights". For each of the attributes CN, gidNumber, homeDirectory, loginShell, uidNumber, Member, and uniqueID do:
    - Press the "Add Property..." button.
    - Locate the attribute in question and select it. You will have to check the "Show all properties" checkbox to see all required attributes. Also note that the sorting order in the dialog is a bit peculiar - attributes that begin with lowercase are sorted after all attributes that begin with uppercase. The easiest way to find attributes is probably to type their name, since the dialog then will find them for you.
    - After selecting the attribute, return to the "Rights assigned to" dialog box by pressing OK, and check the "inheritable" checkbox for the newly added attribute.

Repeat the procedure for [Entry Rights] which is not a normal LDAP attribute but a special keyword. Without browse rights on entries, the user will not be able to see any objects at all which is the first step in reading the information in them.

If **tl-nds-mountpath** (described in Section 10.2.4.4.2.2 is to be used, read access must also be enabled for the `ndsHomeDirectory` attribute.

- Close all dialogs by pressing "OK". Your `posixsearcher` user should now be able to search the directory and retrieve all relevant attributes. Test this by executing the following command:

```
[root@test root]
    ldapsearch -x -D \
    cn=tl-posixsearcher,ou=thinlinc,o=example' -W -H \
    ldaps://ldap.example.com -b \
    ou=People,o=example
```

The output should contain CN, gidNumber, homeDirectory, loginShell, uidNumber, and uid. Also, if you add this users dn and password to `/etc/ldap.conf` on the ldap server, all users should be present in the output of **getent passwd**.

An alternative way of assigning the ACLs required is to add the following LDIF to the toplevel object (`ou=People,o=example` in our example) using `ldapmodify` or a similar tool:

```
ACL: #3#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#CN
```

```
ACL: 3#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#UID
ACL: 3#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#gidNumber
ACL: 3#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#homeDirectory
ACL: 3#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#loginShell
ACL: 3#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#uidNumber
ACL: 3#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#member
ACL: 1#subtree#cn=tl-posixsearcher,ou=thinlinc,o=example#[Entry Rights]
```

## C.6. Creating the DN used to modify users in the directory

In order to function properly, **tl-nds-posixuser** and **tl-nds-posixgroup** needs to bind to eDirectory as a user that has appropriate permissions. Specifically, the user needs to write the attributes uidNumber, gidNumber, loginShell, uniqueID, and homeDirectory. Since an objectclass is also added, the user also needs write access to the objectclass attribute. It must also be able to read the cn attribute, and needs browse permissions on [Entry Rights] in order to find the users at all.

Follow the instructions in Section C.5 but use another username. On the container that is above all users, set "write" permission on uidNumber, gidNumber, loginShell, uniqueID, objectclass and homeDirectory. Set read permission on the cn attribute and browse on [Entry Rights]. The raw ACL list for this looks as follows (given a DN of cn=tl-posixsetter,ou=thinlinc,o=example):

```
ACL: 7#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#objectClass
ACL: 1#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#[Entry Rights]
ACL: 7#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#loginShell
ACL: 7#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#uidNumber
ACL: 7#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#gidNumber
ACL: 7#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#homeDirectory
ACL: 7#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#uid
ACL: 3#subtree#cn=tl-posixsetter,ou=thinlinc,o=example#cn
```

## Appendix D. Configuring CUPS queues on Windows Terminal Servers

If your ThinLinc cluster uses a Windows Terminal Server for some applications, printers must be made available on the Windows servers as well. This is normally handled automatically by ThinLinc, see Section 5.6. In some special cases, it may be necessary to add printers manually. This appendix describes how.

The Nearest Printer and Local Printers are added by defining printer queues that use the Internet Printing Protocol. Create one Network Printer in the printer wizard for each of them, and provide an IPP URL of the following form:

```
http://server_host_name:631/printers/printername
```

For the nearest printer, and following our example with the VSM Server host having a hostname of *tl.example.com*, the URL would be:

```
http://tl.example.com:631/printers/nearest
```

For the local printer, the URL would instead be:

```
http://tl.example.com:631/printers/thinlocal
```

**Note:** Use the hostname of the VSM Server host. The *thinlocal* backend on the VSM Server host will then reroute the *thinlocal* job to the appropriate VSM Agent host.

Use a generic PostScript printer driver for both the *nearest* and the *thinlocal* queues. We recommend the driver "Generic/MS Publisher Imagesetter", which is included in modern Windows versions.

Define all other printer queues using the same method. Either print via CUPS on the VSM Server host, or use an existing printer server if available.

