

# **SSH Tectia® Client 6.1**

## **User Manual**

**30 November 2009**

---

# SSH Tectia® Client 6.1: User Manual

30 November 2009

Copyright © 1995–2009 SSH Communications Security Corp.

This software is protected by international copyright laws. All rights reserved. ssh® and Tectia® are registered trademarks of SSH Communications Security Corp in the United States and in certain other jurisdictions. The SSH and Tectia logos are trademarks of SSH Communications Security Corp and may be registered in certain jurisdictions. All other names and marks are property of their respective owners.

No part of this publication may be reproduced, published, stored in an electronic database, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, for any purpose, without the prior written permission of SSH Communications Security Corp.

THERE IS NO WARRANTY OF ANY KIND FOR THE ACCURACY OR USEFULNESS OF THIS INFORMATION EXCEPT AS REQUIRED BY APPLICABLE LAW OR EXPRESSLY AGREED IN WRITING.

For Open Source Software acknowledgements, see appendix *Open Source Software License Acknowledgements* in the *Product Description*.

SSH Communications Security Corp.

Valimotie 17, FI-00380 Helsinki; Finland

---

# Table of Contents

<b>1. About This Document</b>	7
1.1. Documentation Conventions	8
1.1.1. Operating System Names	8
1.1.2. Directory Paths	9
1.2. Customer Support	9
1.3. Component Terminology	10
<b>2. Installing SSH Tectia Client</b>	13
2.1. Preparing for Installation	13
2.1.1. System Requirements	13
2.1.2. Hardware and Disk Space Requirements	17
2.1.3. Licensing	17
2.1.4. Installation Packages	17
2.1.5. Upgrading Previously Installed SSH Tectia Client Software	19
2.1.6. Downloading SSH Tectia Releases	20
2.2. Installing the SSH Tectia Client Software	21
2.2.1. Installing on AIX	21
2.2.2. Installing on HP-UX	22
2.2.3. Installing on Linux	23
2.2.4. Installing on Solaris	23
2.2.5. Installing on Windows	25
2.2.6. Installing on VMware ESX	28
2.2.7. Installing on Linux on IBM System z	29
2.3. Removing the SSH Tectia Client Software	30
2.3.1. Removing from AIX	30
2.3.2. Removing from HP-UX	31
2.3.3. Removing from Linux	31
2.3.4. Removing from Solaris	31
2.3.5. Removing from Windows	32
2.3.6. Removing from VMware ESX	32
2.3.7. Removing from Linux on IBM System z	33
2.4. Files Related to SSH Tectia Client	33

2.4.1. File Locations on Unix .....	33
2.4.2. File Locations on Windows .....	34
2.4.3. Registry Keys on Windows .....	36
2.5. Symlinks between ssh/scp/sftp and sshg3/scpg3/sftpg3 (on Unix) .....	36
<b>3. Getting Started with SSH Tectia Client .....</b>	<b>39</b>
3.1. Product Components .....	39
3.2. First Login to a Remote Host .....	39
3.2.1. Logging in with SSH Tectia Terminal GUI (on Windows) .....	40
3.2.2. Logging in with Command-Line sshg3 .....	42
3.3. Using Public-Key Authentication .....	44
<b>4. Configuring SSH Tectia Client .....</b>	<b>45</b>
4.1. Configuration Files .....	45
4.1.1. Editing the Configuration Files .....	46
4.2. Command-Line Options .....	46
<b>5. Authentication .....</b>	<b>47</b>
5.1. Server Authentication with Public Keys .....	48
5.1.1. Host Key Storage Formats .....	49
5.1.2. Using the System-Wide Host Key Storage .....	50
5.1.3. Resolving Hashed Host Keys .....	52
5.1.4. Using the OpenSSH known_hosts File .....	53
5.2. Server Authentication with Certificates .....	53
5.2.1. Using the Configuration File (Unix) .....	54
5.2.2. Using the GUI .....	55
5.3. User Authentication with Passwords .....	55
5.3.1. Using the Configuration File (Unix) .....	55
5.3.2. Using Stored Passwords in Connection Profiles .....	56
5.3.3. Using the GUI .....	58
5.4. User Authentication with Public Keys .....	58
5.4.1. Creating Keys with ssh-keygen-g3 .....	59
5.4.2. Uploading Public Keys Manually .....	60
5.4.3. Creating Keys with the Public-Key Authentication Wizard (Windows) .....	63
5.4.4. Using Keys Generated with OpenSSH .....	67
5.4.5. Special Considerations with Windows Servers .....	67
5.5. User Authentication with Certificates .....	67
5.5.1. Using the Configuration File (Unix) .....	68
5.5.2. Using the GUI .....	69
5.6. Host-Based User Authentication (Unix) .....	69
5.7. User Authentication with Keyboard-Interactive .....	69
5.7.1. Using the Configuration File (Unix) .....	70
5.7.2. Using the GUI .....	70
5.8. User Authentication with GSSAPI .....	70
5.8.1. Using the Configuration File (Unix) .....	71
5.8.2. Using the GUI .....	71

<b>6. Transferring Files</b>	73
6.1. Secure File Transfer with <b>scp3</b> and <b>sftp3</b> Commands	73
6.1.1. Using <b>scp3</b>	74
6.1.2. Using <b>sftp3</b>	74
6.2. Secure File Transfer GUI (Windows)	75
6.2.1. Defining File Transfer GUI Settings	75
6.2.2. Downloading Files with the File Transfer GUI	75
6.2.3. Uploading Files with the File Transfer GUI	76
6.2.4. Transfer and Queue Tabs	76
6.2.5. Defining File Properties	77
6.2.6. Differences from Windows Explorer	78
6.3. Controlling File Transfer	79
6.3.1. Site Command	79
<b>7. Secure Shell Tunneling</b>	91
7.1. Local Tunnels	91
7.1.1. Transparent TCP Tunneling	93
7.1.2. Non-Transparent TCP Tunneling	98
7.1.3. Non-Transparent FTP Tunneling	100
7.1.4. SOCKS Tunneling	103
7.2. Remote Tunnels	104
7.3. X11 Forwarding	106
7.4. Agent Forwarding	106
<b>8. Troubleshooting SSH Tectia Client</b>	109
8.1. Starting Connection Broker in Debug Mode	109
8.2. Collecting System Information for Troubleshooting	110
8.3. Answers to Common Problems	111
<b>A. Connection Broker Configuration Tools</b>	115
A.1. SSH Tectia Connections Configuration GUI	115
A.1.1. Opening the GUI	115
A.1.2. Defining General Settings	116
A.1.3. Defining Connection Profiles	128
A.1.4. Defining User Authentication	149
A.1.5. Defining Server Authentication	152
A.1.6. Defining Transparent Tunnels	160
A.1.7. Defining Automatic Tunnels	166
A.2. Configuration File for Connection Broker	168
A.3. Backup of Configuration Files	206
A.4. Broker Configuration File Syntax	207
A.5. SSH Tectia Shortcut Menu (Windows)	216
A.5.1. SSH Tectia Status Dialog Box (Windows)	217
<b>B. Configuring SSH Tectia Terminal and File Transfer GUI (Windows)</b>	221
B.1. Defining Global Settings	221
B.1.1. Defining the Appearance	222

B.1.2. Selecting the Font and Terminal Window Size .....	224
B.1.3. Selecting Colors .....	226
B.1.4. Defining Messages .....	228
B.1.5. Defining File Transfer Settings .....	229
B.1.6. Defining Advanced File Transfer Options .....	233
B.1.7. Defining File Transfer Mode .....	236
B.1.8. Defining Local Favorites .....	237
B.1.9. Defining Security Settings .....	238
B.1.10. Printing .....	239
B.2. Using Command-Line Options .....	241
B.3. Customizing the User Interface .....	242
B.3.1. Saving Settings .....	242
B.3.2. Loading Settings .....	243
B.3.3. Customize Dialog .....	243
B.3.4. Customizing Toolbars .....	248
C. Command-Line Tools and Man Pages .....	249
ssh-broker-g3 .....	250
ssh-broker-ctl .....	255
ssh-troubleshoot .....	259
sshg3 .....	261
scpg3 .....	271
sftpg3 .....	284
ssh-translation-table .....	311
ssh-keygen-g3 .....	316
ssh-keyfetch .....	321
ssh-cmpclient-g3 .....	324
ssh-scepclient-g3 .....	331
ssh-certview-g3 .....	334
ssh-ekview-g3 .....	338
D. Egrep Syntax .....	341
D.1. Egrep Patterns .....	341
D.2. Escaped Tokens for Regex Syntax Egrep .....	342
D.3. Character Sets For Egrep .....	343
E. Audit Messages .....	345
Index .....	369

# Chapter 1 About This Document

This document describes installing and using SSH Tectia Client and the client-side tools of SSH Tectia Server for Linux on IBM System z. In this manual, we use the name SSH Tectia Client to refer to both product versions, as their behaviour is identical.

This manual is meant for SSH Tectia Client users and administrators who install and configure the software.

This document contains the following information:

- Installing SSH Tectia Client and the client-side tools of SSH Tectia Server for Linux on IBM System z
- Getting started
- Configuring SSH Tectia Client
- Authentication
- Transferring files
- Tunneling
- Troubleshooting
- Appendices, including command-line tool, GUI, and audit message references

The Connection Broker handles all cryptographic operations and authentication-related tasks for SSH Tectia Client. In addition, SSH Tectia Client is configured through the Connection Broker settings made either in an XML file, or in the SSH Tectia Configuration GUI as described in [Section A.1](#).

For general information on SSH Tectia Client and its features, refer to *SSH Tectia Client/Server Product Description*.

If you are familiar with SSH Tectia Client 4.x or older, we recommend that you read *SSH Tectia Client/Server Migration Guide*. It contains information on new and changed configuration options of SSH Tectia Client and Server and instructions for migrating existing 4.x installations to version 6.x.

## 1.1 Documentation Conventions

The following typographical conventions are used in SSH Tectia documentation:

**Table 1.1. Documentation conventions**

Convention	Usage	Example
<b>Bold</b>	Menus, GUI elements, strong emphasis	Click <b>Apply</b> or <b>OK</b> .
→	Series of menu selections	Select File → Save
Monospace	Filenames, commands, directories, URLs etc.	Refer to <code>readme.txt</code>
<i>Italics</i>	Reference to other documents or products, emphasis	See <i>SSH Tectia Client User Manual</i>
#	In front of a command, # indicates that the command is run as a privileged user (root).	<pre># rpm --install package.rpm</pre>
\$	In front of a command, \$ indicates that the command is run as a non-privileged user.	<pre>\$ sshg3 user@host</pre>
\	At the end of a line in a command, \ indicates that the command continues on the next line, but there was not space enough to show it on one line.	<pre>\$ ssh-keygen-g3 -t rsa \ -F -c mykey</pre>



### Note

A Note indicates neutral or positive information that emphasizes or supplements important points of the main text. Supplies information that may apply only in special cases (for example, memory limitations, equipment configurations, or specific versions of a program).



### Caution

A Caution advises users that failure to take or to avoid a specified action could result in loss of data.

### 1.1.1 Operating System Names

When the information applies to several operating systems versions, the following naming systems are used:

- **Unix** refers to the following supported operating systems:
  - HP-UX
  - IBM AIX
  - Red Hat Linux, SUSE Linux
  - Linux on IBM System z



- Sun Solaris
- IBM z/OS, when applicable; as SSH Tectia Server for IBM z/OS is running in USS and uses Unix-like tools.
- **z/OS** is used for IBM z/OS, when the information is directly related to IBM z/OS versions.
- **Windows** refers to all supported Windows versions.

## 1.1.2 Directory Paths

The following conventions are used in the documentation to refer to directory paths:

### \$HOME

A Unix environment variable, that indicates the path to the user's home directory.

### %APPDATA%

A Windows environment variable, that indicates the path to the user-specific Application Data folder. By default expands to:

"C:\Documents and Settings\<username>\Application Data" on pre-Vista Windows versions

"C:\Users\<username>\AppData\Roaming" on Windows Vista.

### %USERPROFILE%

A Windows environment variable, that indicates the path to the user-specific profile folder. By default expands to:

"C:\Documents and Settings\<username>" on pre-Vista Windows versions

"C:\Users\<username>" on Windows Vista.

### <INSTALLDIR>

Indicates the default installation directory on Windows:

"C:\Program Files\SSH Communications Security\SSH Tectia"

## 1.2 Customer Support

All SSH Tectia product documentation is available at <http://www.ssh.com/support/documentation/>.

If the product documentation does not answer all your questions, you can find the SSH Tectia FAQ and Knowledge Base at <https://support.ssh.com/>.

If you have purchased a maintenance agreement, you are entitled to technical support from SSH Communications Security. Review your agreement for specific terms and log in at <https://support.ssh.com/>.

Information on submitting support requests, feature requests, or bug reports, and on accessing the online resources is available at <http://www.ssh.com/support/contact/>.

## 1.3 Component Terminology

The following terms are used throughout the documentation.

client computer	The computer from which the Secure Shell connection is initiated.
Connection Broker	The Connection Broker is a component included in SSH Tectia Client, SSH Tectia ConnectSecure, and SSH Tectia MFT Events as well as in the SSH Tectia Server for IBM z/OS client tools. Connection Broker handles all cryptographic operations and authentication-related tasks.
event	An event is a scheduled file transfer or command action pre-configured in the SSH Tectia MFT Events configuration. Events are run automatically according to the defined triggers and conditions.
file transfer GUI	SSH Tectia Client and ConnectSecure include a separate graphical user interface (GUI) for handling and performing file transfers interactively.
host key pair	A public-key pair used to identify a Secure Shell server. The private key file is accessible only to the server. The public key file is distributed to users connecting to the server.
remote host	Refers to the other party of the connection, <a href="#">client computer</a> or <a href="#">server computer</a> , depending on the viewpoint.
Secure Shell client	A client-side application that uses the Secure Shell version 2 protocol, for example <code>sshg3</code> , <code>sftpg3</code> , or <code>scp3</code> of SSH Tectia Client.
Secure Shell server	A server-side application that uses the Secure Shell version 2 protocol.
server computer	The computer on which the Secure Shell service is running and to which the Secure Shell client connects.
SFTP server	A server-side application that provides a secure file transfer service as a subsystem of the Secure Shell server.
SSH Tectia Client	A software component installed on a workstation. SSH Tectia Client provides secure interactive file transfer and terminal client functionality for remote users and system administrators to access and manage servers running SSH Tectia Server or other applications using the Secure Shell protocol. It also supports (non-transparent) static tunneling, and as an optional feature on Windows, transparent TCP tunneling.

SSH Tectia client/server solution	The SSH Tectia client/server solution consists of SSH Tectia Client, SSH Tectia ConnectSecure, SSH Tectia Server, and SSH Tectia Server for IBM z/OS.
SSH Tectia Connections Configuration interface	SSH Tectia Client, ConnectSecure, and Events have a user interface for configuring the connection settings to remote servers.
SSH Tectia ConnectSecure	A software component installed on a server host, but it acts as a Secure Shell client. SSH Tectia ConnectSecure is designed for FTP replacement and it provides FTP-SFTP conversion, transparent FTP tunneling, transparent TCP tunneling, and enhanced file transfer services. SSH Tectia ConnectSecure is capable of connecting to any standard Secure Shell server.
SSH Tectia MFT Events	A software component typically installed on a server host. SSH Tectia MFT Events is designed for automating file transfer and command events and for viewing their performance. The events can be created and maintained via the SSH Tectia MFT Events administration interface. SSH Tectia MFT Events is capable of connecting to any standard Secure Shell server.
SSH Tectia SFTP API	SSH Tectia ConnectSecure includes separate application programming interfaces (API) for C and Java. The APIs can be used by developers who develop secure file transfer applications or integrate SSH Tectia products into other systems.
SSH Tectia Server	SSH Tectia Server is a server-side component where Secure Shell clients connect to. There are three versions of the SSH Tectia Server product available: <i>SSH Tectia Server</i> for Linux, Unix and Windows platforms, <i>SSH Tectia Server for Linux on IBM System z</i> , and <i>SSH Tectia Server for IBM z/OS</i> .
SSH Tectia Server for IBM z/OS	SSH Tectia Server for IBM z/OS provides normal Secure Shell connections and supports the enhanced file transfer (EFT) features and transparent TCP tunneling on IBM mainframes.
SSH Tectia Server for Linux on IBM System z	SSH Tectia Server for Linux on IBM System z provides Secure Shell connections on Linux running on IBM System z platforms.
transparent FTP tunneling	An FTP connection transparently encrypted and secured by a Secure Shell tunnel.
transparent TCP tunneling	A TCP application connection transparently encrypted and secured by a Secure Shell tunnel.
tunneled application	A TCP application secured by a Secure Shell connection.

user key pair

A public-key pair used to identify a Secure Shell user. The private key file is accessible only to the user. The public key file is copied to the servers the user wants to connect to.

## Chapter 2 Installing SSH Tectia Client

This chapter gives instructions on installing (and removing) SSH Tectia Client for each supported platform, and lists the locations of the SSH Tectia files.

### 2.1 Preparing for Installation

This section lists the supported platforms and the pre-requisites for the SSH Tectia Client installation.

#### 2.1.1 System Requirements

Check the following table for the operating systems supported as SSH Tectia Client platforms:

**Table 2.1. Supported operating systems per SSH Tectia product**

Operating System	Client	ConnectSecure	Server	Server for z/Linux <sup>1</sup>	Server for z/OS <sup>1</sup>
HP-UX (PA-RISC)	11i v1, 11i v2, 11i v3	11i v1, 11i v2, 11i v3	11i v1, 11i v2, 11i v3		
HP-UX (IA-64)	11i v2, 11i v3	11i v2, 11i v3	11i v2, 11i v3		
IBM AIX (POWER)	5.3, 6.1	5.3, 6.1 <sup>2</sup>	5.3, 6.1		
Red Hat Enterprise Linux (x86 and x86-64)	4, 5, 5.1, 5.2, 5.3	4, 5, 5.1, 5.2, 5.3	4, 5, 5.1, 5.2, 5.3		
Sun Solaris (SPARC)	9, 10	9, 10	9, 10		
Sun Solaris (x86-64)	10	10	10		
SUSE LINUX Enterprise Desktop (x86 and x86-64)	10	10	10		
SUSE LINUX Enterprise Server (x86)	9, 10	9, 10	9, 10		
SUSE LINUX Enterprise Server (x86-64)	10	10	10		
Microsoft Windows (x86)	XP, Server 2003, Server 2008, Vista, 7 Ultimate	XP, Server 2003	XP, Server 2003, Server 2008		
Microsoft Windows (x64)	Server 2003, Server 2008, Vista, 7 Ultimate		Server 2003, Server 2008		
VMware ESX Server <sup>3</sup>	3.5		3.5		
Red Hat Enterprise Linux for IBM System z (64-bit) <sup>4</sup>				5.1	
SUSE LINUX Enterprise Server for IBM System z (64-bit) <sup>4</sup>				9, 10	
SUSE LINUX Enterprise Server for IBM System z (31-bit)				9	
IBM z/OS (zSeries)					1.8, 1.9, 1.10

<sup>1</sup> Including the client components<sup>2</sup> SSH Tectia ConnectSecure requires reconfiguring the RBAC mode on AIX 6.1.<sup>3</sup> SSH Tectia products can be installed directly on the server virtualization layer.<sup>4</sup> Requires the 31-bit compatibility library `libc.so.6`.



## Note

Keep the operating system fully patched according to recommendations by the operating system vendor.

The supported operating systems are required to have the following or superseding patches or maintenance levels installed. The listed patches have been tested with SSH Tectia.

- IBM AIX 5L 5.3: maintenance level 5
- Microsoft Windows XP: Service Pack 2
- Microsoft Windows Server 2003: Service Pack 2.
- Microsoft Windows Server 2008: Service Pack 1.
- Red Hat Enterprise Linux 4 and 5: xorg-x11-xauth is required for X11 forwarding
- Sun Solaris 9:
  - The Solaris 9 Recommended Patch Cluster, Jun 5, 2006
  - 112908-30 krb5, gss Patch, May 30, 2007
  - 111711-08 32-bit Shared library patch for C++, Jan, 2004
  - 111712-25 64-bit Shared library patch for C++, Nov, 2009
- SUSE Linux Enterprise Server 10: Service pack 1
- SUSE Linux Enterprise Server 9 for IBM System z: Service pack 4
- SUSE Linux Enterprise Server 10 for IBM System z: Service pack 1

## HP-UX patches

The general principle is to install the latest **HP-required patch bundle** for the OS version, currently required bundles exist for 11i v1 and 11i v2. Proper functioning of the SSH Tectia software also requires the latest **HP recommended patches** for libc, pthread and linker tools. In addition, some individual patches may be needed to fix specific problems. Such patches are mentioned separately.



## Note

Check the HP web-site for any newer patches superseding the ones listed here. We recommend installing the latest version recommended by HP.

- HP-UX 11i v1 on PA-RISC requires the following patches:
  - B.11.11.0306.1 patch bundle for HP-UX 11i v1, Jun 2003

- PHKL\_34738 signal cumulative patch, Sep 2006
- PHKL\_34173 select(2) delay, hang, Jan 2006
- PHKL\_28122 signals, threads enhancement, psets enablement, Feb 2003
- PHNE\_36576 cumulative STREAMS patch, Apr 2008
- PHNE\_38678 cumulative ARPA transport patch, Apr 2009

Install also these (currently) latest HP-recommended patches for libc, pthread and ld(1) and linker tools:

- PHCO\_33282 pthread library cumulative patch, Oct 2006
- PHCO\_35743 libc cumulative patch, Jan 2007
- PHSS\_37516 ld(1) and linker tools cumulative patch, Dec 2007
- HP-UX 11i v2 on PA-RISC and IA-64 (Itanium):
  - B.11.23.0409.3 patch bundle for HP-UX 11i v2, Sep 2004
  - PHCO\_34191 libc cumulative patch, Mar 2003
  - PHCO\_36323 pthread library cumulative patch, Aug 2007
  - PHSS\_37492 linker + fdp cumulative patch, Dec 2007
  - Kerberos Client D.1.6.2, Dec 2007
  - PHNE\_34788 cumulative STREAMS patch, May 2007
  - PHNE\_37395 cumulative ARPA transport patch, Dec 2007

For X11 forwarding, install also the following patches on the server machine you want to forward X11

- PHSS\_34159 XClients patch, feb 2006
- PHSS\_35046 XMotif Runtime patch, Oct 2006
- HP-UX 11i v3 on PA-RISC and IA-64 (Itanium):
  - PHCO\_36551 pthread library cumulative patch, May 2007
  - PHSS\_37202 linker and fdp cumulative patch, Oct 2007
  - Kerberos Client D.1.6.2, Dec 2007



## 2.1.2 Hardware and Disk Space Requirements

SSH Tectia Client does not have any special hardware requirements. Any computer capable of running a current version of the listed operating systems, and equipped with a functional network connection can be used.

The SSH Tectia Client installation requires about 100 megabytes of disk space.

Note that SSH Tectia Client will save each user's settings in that particular user's personal directory.

## 2.1.3 Licensing

SSH Tectia Client requires a license to function. The license file is named `stc61.dat`

Depending on the platform for which you have purchased SSH Tectia Client, consider the following license-related issues:

- On Unix, you need to install the license file manually to directory: `/etc/ssh2/licenses`
- On Windows, the installation wizard automatically copies the license file to the correct directory when installing from the CD-ROM or from an extracted online package.

After installation, the license file is located in the default installation directory:

```
"C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia AUX\licenses"
```

- On the CD-ROM, the license files can be found in directory: `install/<platform>`
- In the online installation packages, the license files are included in the compressed files (zip/tar) together with the `releasenotes.txt` files and the PDF-format documentation.
- Evaluation packages have a temporary license for 45 days. On Unix machines, a banner message will remind users of how many days are left until the license expires.

## 2.1.4 Installation Packages

The installation packages of SSH Tectia Client are compressed into installation bundles. There are two bundles for each supported operating system, the commercial version (`-comm`) and the upgrade and evaluation version (`-upgrd-eval`). The evaluation versions can be used as upgrade packages, if you already have a suitable license.

Select the relevant SSH Tectia Client bundle:

- For AIX platforms:

```
ssh-tectia-client-<version>-aix-5-6-powerpc-comm.tar  
ssh-tectia-client-<version>-aix-5-6-powerpc-upgrd-eval.tar
```

- For HP-UX PA-RISC platforms:

```
ssh-tectia-client-<version>-hpux-11i-hppa-comm.tar  
ssh-tectia-client-<version>-hpux-11i-hppa-upgrd-eval.tar
```

- For HP-UX Itanium platforms:

```
ssh-tectia-client-<version>-hpux-11i-ia64-comm.tar  
ssh-tectia-client-<version>-hpux-11i-ia64-upgrd-eval.tar
```

- For Linux 32-bit platforms:

```
ssh-tectia-client-<version>-linux-x86-comm.tar  
ssh-tectia-client-<version>-linux-x86-upgrd-eval.tar
```

- For Linux 64-bit platforms:

```
ssh-tectia-client-<version>-linux-x86_64-comm.tar  
ssh-tectia-client-<version>-linux-x86_64-upgrd-eval.tar
```

- For Solaris SPARC platforms:

```
ssh-tectia-client-<version>-solaris-9-10-sparc-comm.tar  
ssh-tectia-client-<version>-solaris-9-10-sparc-upgrd-eval.tar
```

- For Solaris x86-64 platforms:

```
ssh-tectia-client-<version>-solaris-10-x86_64-comm.tar  
ssh-tectia-client-<version>-solaris-10-x86_64-upgrd-eval.tar
```

- For Windows platforms:

```
ssh-tectia-client-<version>-windows-comm.tar  
ssh-tectia-client-<version>-windows-upgrd-eval.tar
```

- For VMware ESX platforms:

```
ssh-tectia-client-<version>-linux-x86-comm.tar  
ssh-tectia-client-<version>-linux-x86-upgrd-eval.tar
```

- For Linux on IBM System z platforms (31-bit and 64-bit), the client-side tools are included in the following installation bundle:

```
ssh-tectia-server-ibm-z-<version>-linux-s390-comm.tar  
ssh-tectia-server-ibm-z-<version>-linux-s390-upgrd-eval.tar
```

<version> indicates the product release and the current build number (for example 6.1.4.123).

Inside the installation bundles are the actual installation packages for SSH Tectia Client. Select the packages to install according to which product features are relevant in your environment.

On Unix and Linux platforms, the SSH Tectia Client comes in three installation packages:

- the `ssh-TECTIA-common` package contains the common components of SSH Tectia Client and Server.
- the `ssh-TECTIA-client` package contains the specific components of SSH Tectia Client.
- the *optional* `ssh-TECTIA-guisupport` package contains the components required for the GUI available on Linux platforms.

On Windows, SSH Tectia Client comes in a single MSI installation package, and the installation wizard guides you to select which components to install.

## 2.1.5 Upgrading Previously Installed SSH Tectia Client Software



### Note

Before starting the upgrade, make backups of all configuration files where you have made modifications. See instructions in [Section A.3](#).

If you are running both SSH Tectia client-side tools and SSH Tectia Server on the same machine, install the same release of each SSH Tectia product, because there are dependencies between the common components.

Check if you have some Secure Shell software, for example earlier versions of SSH Tectia products or OpenSSH server or client, running on the machine where you are planning to install the new SSH Tectia versions.

Before installing SSH Tectia Server on Unix platforms, stop any OpenSSH servers running on port 22, or change their listener port. You do not need to uninstall the OpenSSH software.

The following table shows you which SSH Tectia versions you need to uninstall before you can upgrade to SSH Tectia Client 6.1. When upgrading versions marked *upgrade on top*, the earlier version is automatically removed during the upgrade procedure. Also, when installing via SSH Tectia Manager, the earlier versions are removed automatically.

**Table 2.2. Upgrade lines**

SSH Tectia version	AIX	HP-UX	Linux	Solaris	Windows
SSH Secure Shell 2.x-3.x	remove	remove	remove	remove	remove
SSH Tectia 4.0	remove	remove	remove	remove	remove
SSH Tectia 4.1-4.x	remove	remove	remove	remove	upgrade on top
SSH Tectia 5.x-6.x	upgrade on top	upgrade on top	upgrade on top	remove	upgrade on top

The configuration file format and file locations have been changed in SSH Tectia Client 5.0. Because of that, the configuration files behave differently when upgrading from 4.x and from 5.x-6.x:

- The 4.x configuration files are *not* migrated to 6.1, but the default 6.1 configuration is used. However, the connection profiles are migrated from 4.x to 6.1 on Windows platforms.
- The 5.x-6.x configuration files are migrated to 6.1 as such and automatically taken into use there.

When necessary, you can modify the configuration files by using the SSH Tectia Configuration GUI or by editing the XML configuration files manually with an ASCII text editor or an XML editor.

A separate document, *SSH Tectia Client/Server Migration Guide*, gives detailed instructions on upgrading from the SSH Tectia client/server solution 4.x to 6.x, including information on migrating the configuration files.

On Windows, a backup copy is automatically made of the earlier SSH Tectia Client configuration files and stored in the user-specific directory:

```
"%APPDATA%\SSH\backup-<version>-<date>"
```

where

<version> is the SSH Tectia release

<date> is the date of the upgrade.

## 2.1.6 Downloading SSH Tectia Releases

All releases require a commercial license that is delivered with the installation package.

To download SSH Tectia software from the SSH Customer Download Center:

1. Log in to customer download center at: <https://downloads.ssh.com>
2. Select **SSH Tectia Client** from the navigator list, and choose the relevant version. SSH Tectia products are published in major, minor, and maintenance releases:
  - Major releases are indicated with full numbers, for example 5.0 and 6.0. Major releases publish new products and new major features to existing products, in addition to fixes to the previous versions.
  - Minor releases are indicated with the second digit in the release numbers, for example 5.3 or 6.1. Minor releases publish new features and fixes to the previous versions.
  - Maintenance releases are third digit versions (for example 5.3.6 or 6.0.1). Maintenance releases provide fixes to the previous versions, not new functionality. The maintenance releases are available for customers with Maintenance and Support Agreement.

3. Click the product version, and the compressed installation package will be downloaded to the default download folder on your machine.
4. Proceed to the installation. See the platform-specific installation instructions for SSH Tectia Client below.

## 2.2 Installing the SSH Tectia Client Software

This section gives instructions on installing SSH Tectia Client locally on the supported operating systems.

You can get the installation packages on an installation CD-ROM or you can download them from the SSH Communications Security website as explained in [Section 2.1.6](#).

SSH Tectia Client can also be installed via SSH Tectia Manager. For instructions on this, see *SSH Tectia Manager Administrator Manual*.

See the installation instructions for SSH Tectia Client per platform in the following sections.

### 2.2.1 Installing on AIX

The downloaded online installation package contains the compressed installation files. On the CD-ROM, the installation packages are located in directory `/install/aix/`.

Two packages are required: one for the common components of SSH Tectia Client and Server, and one for the specific components of SSH Tectia Client.

To install SSH Tectia Client on AIX, follow the instructions below:

1. Unpack the installation packages:

```
$ unzip ssh-tectia-common-<version>-aix-5-6-powerpc.bff.Z
$ unzip ssh-tectia-client-<version>-aix-5-6-powerpc.bff.Z
```

In the commands, `<version>` is the current package version of SSH Tectia Client (for example, 6.1.4.123).

2. Install the packages by running the following commands with root privileges:

```
# installp -d ssh-tectia-common-<version>-aix-5-6-powerpc.bff SSHTectia.Common
# installp -d ssh-tectia-client-<version>-aix-5-6-powerpc.bff SSHTectia.Client
```

3. Copy the license file to directory: `/etc/ssh2/licenses`. (*This is not necessary in "third-digit" maintenance updates.*) See also [Section 2.1.3](#).

## 2.2.2 Installing on HP-UX

Check that you have the operating system fully patched. See the latest patch information on the Hewlett-Packard web site. In case PAM/Kerberos is used on a HP-UX platform, install also the latest patches related to Kerberos.

The downloaded online installation package contains the compressed installation files. On the CD-ROM, the installation packages for HP-UX platforms are located in the `/install/hp-ux/` directory.

Two packages are required: one for the common components of SSH Tectia Client and Server, and one for the specific components of SSH Tectia Client.

To install SSH Tectia Client on HP-UX, follow the instructions below:

1. Select the installation package according to your HP-UX version.

When installing on HP-UX 11i v1 (11.11), 11i v2 (11.23), or 11i v3 (11.31) running on the PA-RISC architecture, use the packages named:

```
ssh-tectia-common-<version>-hpux-11i-hppa.depot.Z
ssh-tectia-client-<version>-hpux-11i-hppa.depot.Z
```

When installing on HP-UX 11i v2 or 11i v3 running on the Itanium architecture, use the packages named:

```
ssh-tectia-common-<version>-hpux-11i-ia64.depot.Z
ssh-tectia-client-<version>-hpux-11i-ia64.depot.Z
```

In the commands, `<version>` indicates the product release version and the current build number (for example, 6.1.4.123).

2. Unpack the packages with `uncompress`. In order to be installable, the created packages must have the correct long file name. In the command examples below, we use the Itanium versions:

```
$ uncompress ssh-tectia-common-<version>-hpux-11i-ia64.depot.Z
$ uncompress ssh-tectia-client-<version>-hpux-11i-ia64.depot.Z
```

3. Install the packages by running the following commands with root privileges:

```
# swinstall -s <path>/ssh-tectia-common-<version>-hpux-11i-ia64.depot SSHG3common
# swinstall -s <path>/ssh-tectia-client-<version>-hpux-11i-ia64.depot SSHG3client
```



### Note

In the commands, `<path>` is the full path to the installation package. HP-UX requires the full path even when the command is run in the same directory.

4. Copy the license file to directory: `/etc/ssh2/licenses` (*This is not necessary in "third-digit" maintenance updates.*) See [Section 2.1.3](#).

## 2.2.3 Installing on Linux

SSH Tectia Client for Linux platforms is supplied in RPM (Red Hat Package Manager) binary packages for Red Hat Enterprise Linux and SUSE Linux. There are separate packages for Linux versions running on the 32-bit x86 and the 64-bit x86-64 platform architecture.

The downloaded online installation package contains the RPM installation files. On the installation CD-ROM, the installation packages for Linux are located in the `/install/linux/` directory.

Two packages are always required: one for the common components of SSH Tectia Client and Server, and one for the specific components of SSH Tectia Client. If you want to use the product with a graphical user interface (GUI), install also the optional GUI support package.

To install SSH Tectia Client on Linux, follow the instructions below:

1. Select the installation package according to your Linux architecture.

When installing on SUSE or Red Hat Enterprise Linux running on the 32-bit x86 architecture, use the packages named:

```
ssh-tectia-common-<version>-linux-x86.rpm  
ssh-tectia-client-<version>-linux-x86.rpm  
ssh-tectia-guisupport-<version>-linux-x86.rpm
```

When installing on SUSE or Red Hat Enterprise Linux versions running on the 64-bit x86-64 architecture, use the packages named:

```
ssh-tectia-common-<version>-linux-x86_64.rpm  
ssh-tectia-client-<version>-linux-x86_64.rpm  
ssh-tectia-guisupport-<version>-linux-x86_64.rpm
```

In the commands, `<version>` indicates the product release version and the current build number (for example, 6.1.4.123).

2. Install the packages with root privileges. In the command examples below, we use the x86-64 version:

```
# rpm -Uvh ssh-tectia-common-<version>-linux-x86_64.rpm  
# rpm -Uvh ssh-tectia-client-<version>-linux-x86_64.rpm  
# rpm -Uvh ssh-tectia-guisupport-<version>-linux-x86_64.rpm
```

3. Copy the license file to the `/etc/ssh2/licenses` directory. (*This is not necessary in "third-digit" maintenance updates.*) See also [Section 2.1.3](#).

## 2.2.4 Installing on Solaris

The downloaded online installation package contains the compressed installation files. On the CD-ROM, the installation packages for Solaris are located in directory `/install/solaris/`.

Two packages are required: one for the common components of SSH Tectia Client and Server, and one for the specific components of SSH Tectia Client.

SSH Tectia Client includes support for Zones on Solaris 10. The SSH Tectia software can be installed into the global and local zones. When the SSH Tectia software is installed into the global zone, it becomes automatically installed also into the existing local zones. However, if the local zones are added into the system later, the SSH Tectia Client needs to be separately installed on them.

In case you are installing SSH Tectia Client into a sparse zone, note that the installation process will report a failure in creating symlinks. The actual installation is finished successfully, but you need to manually add the `/opt/tektia/bin` to the path settings.

For information on the Solaris Zones, see the Sun Microsystems documentation: *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*.

To install SSH Tectia Client on Solaris, follow the instructions below:

1. Select the installation package according to your Sun Solaris version.

When installing on Solaris version 9 or 10 running on the SPARC architecture, use the packages named:

```
ssh-tektia-common-<version>-solaris-9-10-sparc.pkg.Z  
ssh-tektia-client-<version>-solaris-9-10-sparc.pkg.Z
```

When installing on Solaris version 10 running on the x86-64 architecture, use the packages named:

```
ssh-tektia-common-<version>-solaris-10-x86_64.pkg.Z  
ssh-tektia-client-<version>-solaris-10-x86_64.pkg.Z
```

In the commands, `<version>` indicates the product release version and the current build number (for example, 6.1.4.123).

2. Unpack the installation packages to a suitable place. The standard place is `/var/spool/pkg` in Solaris environment. In the command examples below, we use Solaris 10 x86-64:

```
$ uncompress ssh-tektia-common-<version>-solaris-10-x86_64.pkg.Z  
$ uncompress ssh-tektia-client-<version>-solaris-10-x86_64.pkg.Z
```

3. Install the packages with the `pkgadd` tool with root privileges:

```
# pkgadd -d ssh-tektia-common-<version>-solaris-10-x86_64.pkg all  
# pkgadd -d ssh-tektia-client-<version>-solaris-10-x86_64.pkg all
```

4. Copy the license file to directory: `/etc/ssh2/licenses` (*This is not necessary in "third-digit" maintenance updates.*) See also [Section 2.1.3](#).



## 2.2.5 Installing on Windows

The Windows installation packages are provided in the MSI (Microsoft Installer) format. The same package is compatible with the supported 32-bit (x86) and the 64-bit (x64) versions of Microsoft Windows.

On the CD-ROM, the installation package for Windows is located in the `/install/windows/` directory.

SSH Tectia Client includes support for Entrust certificates on Windows XP. The necessary libraries are automatically included in the installation.

The online installation package is a zip file containing the license file and the executable Microsoft Installer (MSI) package.

The installation is carried out by a standard installation wizard. The wizard prompts you for information, copies the program files, and sets up the client.

If you are upgrading a previous installation of SSH Tectia Client, see first [Section 2.1.5](#).

To install SSH Tectia Client on Windows, follow the instructions below:

1. Extract the installation zip file contents to a temporary location.



### Note

The license file will be imported automatically, when you extract the contents of the online `.zip` package before running the `.msi` installer, or if you are installing from a CD-ROM.

If you run the `.msi` installer directly from the online `.zip` package, you need to manually import the `stc61.dat` license file after completing the installation. The installation wizard will show an error message about missing license file (see below), and when you attempt to start the SSH Tectia Client, you are prompted to import the license manually to the correct directory:

```
"C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia AUX\licenses"
```

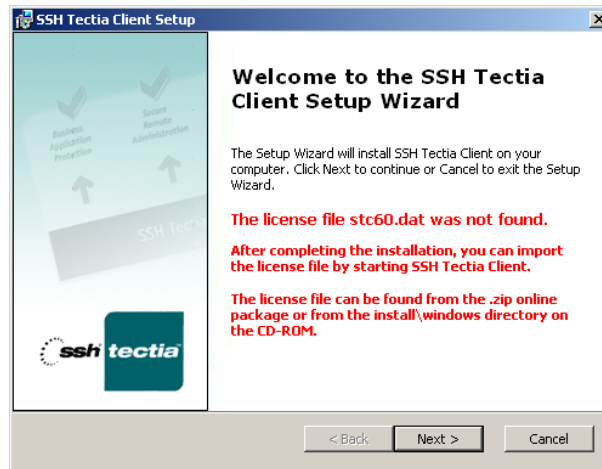


Figure 2.1. Warning about a missing license file

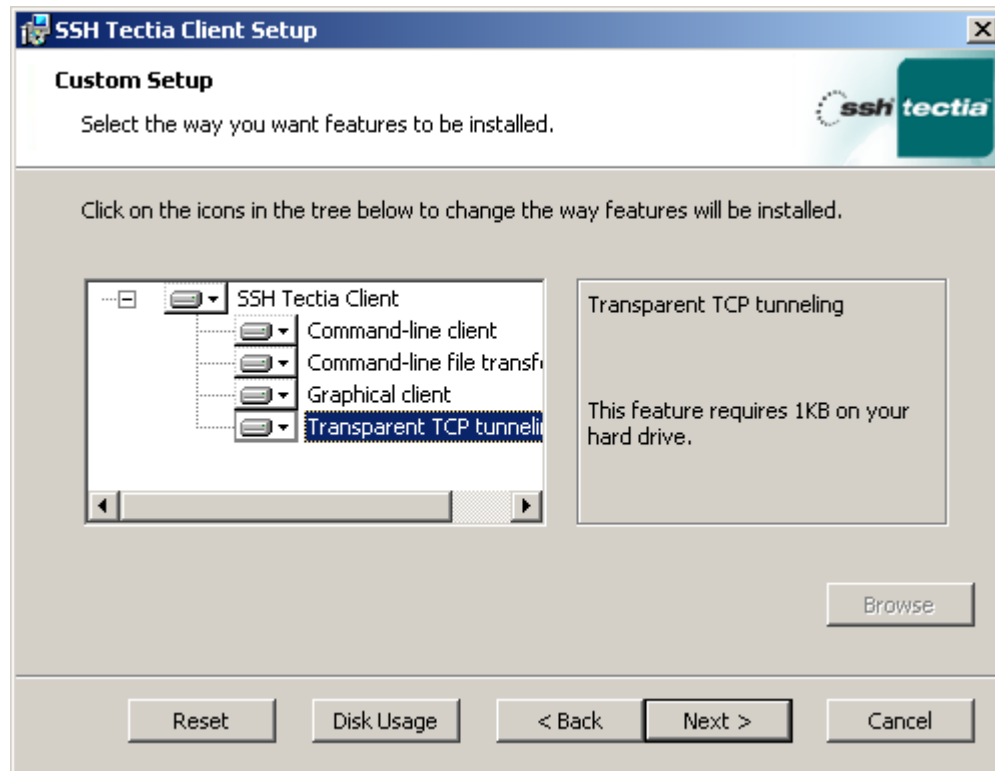
2. Locate the installation file `ssh-tectia-client-<version>-windows.msi` (where `<version>` corresponds to the version and build number, for example `6.1.4.123`).
3. Double-click the installation file, and the installation wizard will start.
4. Follow the wizard through the installation steps and fill in information as requested.
5. The **Typical** installation of SSH Tectia Client includes the `sshg3.exe`, `scpg3.exe`, and `sftpg3.exe` command-line tools, and the graphical user interface for terminal and file transfer.

To install all components, select **Complete** when the wizard prompts for the setup type.

### Note

Transparent TCP tunneling is supported as an optional feature on Windows XP.

If you want to select the components to install, select **Custom** when the wizard prompts for the setup type. The next dialog box allows you to exclude some of the components from the installation. See [Figure 2.2](#).



**Figure 2.2. Installation options with SSH Tectia Client**

6. When the installation has finished, click **Finish** to exit the wizard.

The default installation directory is "C:\Program Files\SSH Communications Security\SSH Tectia".

### 2.2.5.1 Silent Installation

SSH Tectia Client can also be installed silently on a workstation. Silent (non-interactive) installation means that the installation procedure will not display any user interface and will not ask any questions from the user. This option is especially useful for system administrators, as it allows remotely-operated automated installations.

In silent mode, SSH Tectia Client is installed with the default settings and without any additional features.

The following command can be used to install SSH Tectia Client silently:

```
msiexec /q /i ssh-tectia-client-<version>-windows.msi INSTALLDIR="<path>"
```

In the command, <version> is the current version of SSH Tectia Client (for example, 6.1.4.123), and <path> is the path to the desired installation directory. If the INSTALLDIR variable is omitted, SSH Tectia Client is installed to the default location ("C:\Program Files\SSH Communications Security\SSH Tectia").

### 2.2.5.2 SSH Tectia Program Group

The installation creates a new program group in the **Start** → **Programs** menu. The default name for this program group is **SSH Tectia Client**.



**Figure 2.3. The SSH Tectia Client program group**

The **Documentation** link opens the User Documentation view on the SSH website.

The **SSH Tectia - File Transfer** link opens a Secure Shell session with the file transfer GUI.

The **SSH Tectia - Terminal** link opens a Secure Shell session with the terminal GUI.

The **SSH Tectia - Configuration** link opens the GUI for configuring the Connection Broker.

### 2.2.5.3 Desktop Icons

During installation, SSH Tectia icons are added to your desktop. There are separate program icons for SSH Tectia Terminal and File Transfer windows. They both start the same application, `ssh-client-g3.exe`. The first icon starts the terminal window and the latter starts the file transfer window.



**Figure 2.4. The SSH Tectia Terminal icon**



**Figure 2.5. The SSH Tectia File Transfer icon**

## 2.2.6 Installing on VMware ESX

SSH Tectia Client can be installed on VMware ESX in two ways:

- directly on the VMware ESX Server's Service Console; this section gives instructions for this case.
- on a VMware guest host running a supported version of the operating system; see the platform-specific installation instructions above.

SSH Tectia functionality has been tested on VMware ESX Server's Service Console as a replacement of OpenSSH.



## Note

Keep in mind that SSH Tectia Client software may need to be re-installed after every VMware ESX upgrade.

When installing SSH Tectia Client directly on the VMware ESX Service Console, use the SSH Tectia Linux installation package for the x86 platform architecture. The downloaded online installation package contains the RPM installation files. On the installation CD-ROM, the installation packages for Linux are located in the `/install/linux/` directory.

Two packages are always required: one for the common components of SSH Tectia Client and Server, and one for the specific components of SSH Tectia Client. If you want to use the product with a graphical user interface (GUI), install also the optional GUI support package.

To install SSH Tectia Client on the VMware ESX Service Console, follow the instructions below:

1. Download the installation packages:

```
ssh-tectia-common-<version>-linux-x86.rpm  
ssh-tectia-client-<version>-linux-x86.rpm  
ssh-tectia-guisupport-<version>-linux-x86.rpm
```

In the commands, `<version>` indicates the product release version and the current build number (for example, 6.1.4.123).

2. Install the packages with root privileges:

```
# rpm -Uvh ssh-tectia-common-<version>-linux-x86.rpm  
# rpm -Uvh ssh-tectia-client-<version>-linux-x86.rpm  
# rpm -Uvh ssh-tectia-guisupport-<version>-linux-x86.rpm
```

3. Copy the license file to the `/etc/ssh2/licenses` directory. (*This is not necessary in "third-digit" maintenance updates.*) See also [Section 2.1.3](#).

## 2.2.7 Installing on Linux on IBM System z

SSH Tectia Server for Linux on IBM System z includes client-side tools that can be installed the same way as SSH Tectia Client.

The installation files for both SSH Tectia Server for Linux on IBM System z and the client-side tools are included in the same installation bundle called `ssh-TECTIA-server-ibm-z-<version>-linux-s390-comm.tar`. The same installation package can be used on all supported Linux versions on IBM System z. On the installation CD-ROM, the installation packages for Linux on IBM System z are located in the `/install/linux/` directory.

Download the online installation package containing the RPM (Red Hat Package Manager) binary packages. Two packages are required: one for the common components of SSH Tectia Server and the client-side tools, and one for the specific components of the client.

To install the client-side tools of SSH Tectia Server for Linux on IBM System z, follow the instructions below:

1. Select the installation packages for Linux on IBM System z:

```
ssh-TECTIA-common-<version>-linux-s390.rpm  
ssh-TECTIA-client-<version>-linux-s390.rpm
```

In the commands, `<version>` indicates the product release version and the current build number (for example, `6.1.4.123`).

2. Install the packages with root privileges:

```
# rpm -Uvh ssh-TECTIA-common-<version>-linux-s390.rpm  
# rpm -Uvh ssh-TECTIA-client-<version>-linux-s390.rpm
```

3. Copy the license file to the `/etc/ssh2/licenses` directory. See also [Section 2.1.3](#).

## 2.3 Removing the SSH Tectia Client Software

This section gives instructions on removing SSH Tectia Client from the supported operating systems.



### Note

The uninstallation procedure removes only the files that were created when installing the software. Any configuration files have to be removed manually.

### 2.3.1 Removing from AIX

To remove SSH Tectia Client from an AIX environment, follow the instructions below:

1. Remove the installation by issuing the following command with root privileges:

```
# installp -u SShTectia.Client
```

2. If you want to remove also the components that are common with SSH Tectia Server, give the following command:

```
# installp -u SSHTectia.Common
```

Note that removing the common components disables SSH Tectia Server, if it has been installed on the same host.

## 2.3.2 Removing from HP-UX

To remove SSH Tectia Client from an HP-UX environment, follow the instructions below:

1. Remove the installation by issuing the following command with root privileges:

```
# swremove SSHG3client
```

2. If you want to remove also the components that are common with SSH Tectia Server, give the following command:

```
# swremove SSHG3common
```

Note that removing the common components disables SSH Tectia Server, if it has been installed on the same host.

## 2.3.3 Removing from Linux

To remove SSH Tectia Client from a Linux environment, follow the instructions below:

1. Remove the installation by issuing the following command with root privileges:

```
# rpm -e ssh-tectia-client-<version>
```

In the command, <version> is the package version of SSH Tectia Client to be removed (for example, 6.1.4.123).

2. If you want to remove also the components that are common with SSH Tectia Server, give the following command:

```
# rpm -e ssh-tectia-common-<version>
```

Note that removing the common components disables SSH Tectia Server, if it has been installed on the same host.

3. To remove the GUI components, issue the following command with root privileges:

```
# rpm -e ssh-tectia-guisupport-<version>
```

## 2.3.4 Removing from Solaris

To remove SSH Tectia Client from a Solaris environment, follow the instructions below:

1. Remove the installation by issuing the following command with root privileges:

```
# pkgrm SSHG3clnt
```

2. If you want to remove also the components that are common with SSH Tectia Server, give the following command:

```
# pkgrm SSHG3cmmn
```

Note that removing the common components disables SSH Tectia Server, if it has been installed on the same host.

## 2.3.5 Removing from Windows

There are several ways to remove the SSH Tectia Client installation from Windows. Follow one set of instructions below:

### Using Windows Control Panel tools

1. Open the **Control Panel** and double-click the **Add or Remove Programs** option.
2. Select **SSH Tectia Client** from the list of installed programs and click the **Remove** button.
3. Click **Yes** to confirm.

### Using the Microsoft Installer

1. Locate the installation file `ssh-tectia-client-<version>-windows.msi` (where `<version>` corresponds to the version and build number, for example 6.1.4.123).
2. Double-click the msi file, and the Microsoft installer will start.
3. Select **Remove** to commence the uninstallation.
4. Click **Finish** when the removal has been completed.

### Using Silent Command-line Tools

SSH Tectia Client can also be removed silently by giving the following command:

```
msiexec /q /x ssh-tectia-client-<version>-windows.msi
```

In the command, `<version>` is the version of SSH Tectia Client to be removed (for example, 6.1.4.123).

## 2.3.6 Removing from VMware ESX

To remove SSH Tectia Client from a VMware ESX platform, follow the instructions below:

1. Remove the installation by issuing the following command with root privileges:



```
# rpm -e ssh-tectia-client-<version>
```

In the command, <version> is the package version of SSH Tectia Client to be removed (for example, 6.1.4.123).

2. If you want to remove also the components that are common with SSH Tectia Server, give the following command:

```
# rpm -e ssh-tectia-common-<version>
```

Note that removing the common components disables SSH Tectia Server, if it has been installed on the same host.

3. To remove the GUI components, issue the following command with root privileges:

```
# rpm -e ssh-tectia-guisupport-<version>
```

## 2.3.7 Removing from Linux on IBM System z

Removing of the client-side tools of SSH Tectia Server for Linux on IBM System z is done the same way as removing SSH Tectia Client. To remove the client-side tools from Linux on IBM System z, follow the instructions below:

1. Remove the installation by issuing the following command with root privileges:

```
# rpm -e ssh-tectia-client-<version>
```

In the command, <version> is the package version to be removed (for example, 6.1.4.123).

2. If you want to remove also the components that are common with SSH Tectia Server for Linux on IBM System z, give the following command:

```
# rpm -e ssh-tectia-common-<version>
```

Note that removing the common components disables SSH Tectia Server, if it has been installed on the same host.

## 2.4 Files Related to SSH Tectia Client

This section lists the default locations where the installation process will store the SSH Tectia Client executables, configuration files, the license file, and the user-specific configuration files.

### 2.4.1 File Locations on Unix

On Unix platforms, the SSH Tectia Client files are located in the following directories:

- `/etc/ssh2`
- `/etc/ssh2/ssh-broker-config.xml`: the global Connection Broker configuration file (see [ssh-broker-config\(5\)](#))
- `/etc/ssh2/ssh-broker-config-example.xml`: a sample file with Connection Broker configuration examples
- `/etc/ssh2/licenses`: the license file directory (see [Section 2.1.3](#)).
- `/etc/ssh2/hostkeys`: the global directory for known remote host keys
- `/etc/ssh2/ssh-tectia/auxdata/ssh-broker-ng`: the Connection Broker configuration file DTD directory
- `/etc/ssh2/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`: this configuration file is read first, and it holds the factory default settings. **Do not edit** this file, but you can use it to view the default settings. This file must be available and correctly formatted for the Connection Broker to get started. For the configuration options, see [ssh-broker-config\(5\)](#).
- `/opt/tectia/bin`: user binaries such as `sshg3` and `ssh-broker-g3`
- `/opt/tectia/man`: the manual pages
- `/opt/tectia/libexec`: library binaries
- `/opt/tectia/lib/sshsecsh`: library binaries

The user-specific configurations are stored in the following directories:

- `$HOME/.ssh2/ssh-broker-config.xml`: the user-specific Connection Broker configuration file
- `$HOME/.ssh2`: the default directory for user keys
  - `$HOME/.ssh2/random_seed`: the seed file for the random number generator
  - `$HOME/.ssh2/hostkeys`: the user-specific directory for known remote host keys
  - `$HOME/.ssh2/identification`: (*optional*) the identification file used with public-key authentication

## 2.4.2 File Locations on Windows

On Windows, the default installation directory (<INSTALLDIR> below) for SSH Tectia products is:

"C:\Program Files\SSH Communications Security\SSH Tectia"

On Windows, the SSH Tectia Client files are located in the following directories:

- "<INSTALLDIR>\SSH Tectia Client": the binaries for SSH Tectia Client

- "<INSTALLDIR>\SSH Tectia Broker": the Connection Broker binaries and example configuration files
- "<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml": the global Connection Broker configuration file (see its manpage: [ssh-broker-config\(5\)](#))
- "<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config-example.xml": a sample file with Connection Broker configuration examples
- "<INSTALLDIR>\SSH Tectia AUX": auxiliary files and binaries such as ssh-keygen-g3.exe
- "<INSTALLDIR>\SSH Tectia AUX\ssh-events\ssh-events-config-1.dtd": the document type definition (DTD) used with the SSH Tectia MFT Events configuration files. **Do not edit** this file!
- "<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng": the Connection Broker configuration file DTD directory
  - "<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml": this configuration file is read first and it holds the factory default settings. **Do not edit** this file, but you can use it to view the default settings. This file must be available and correctly formatted for the Connection Broker to get started. For the configuration options, see [ssh-broker-config\(5\)](#).
  - "<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-ng-config-1.dtd": the document type definition (DTD) used with the Connection Broker configuration files. **Do not edit** this file!
- "<INSTALLDIR>\SSH Tectia AUX\licenses": the license file directory (see [Section 2.1.3](#))
- "<INSTALLDIR>\SSH Tectia AUX\documents": the end-user license agreements.

Figure 2.6 shows the SSH Tectia directory structure in the Windows Start menu when several SSH Tectia products have been installed on the same machine.

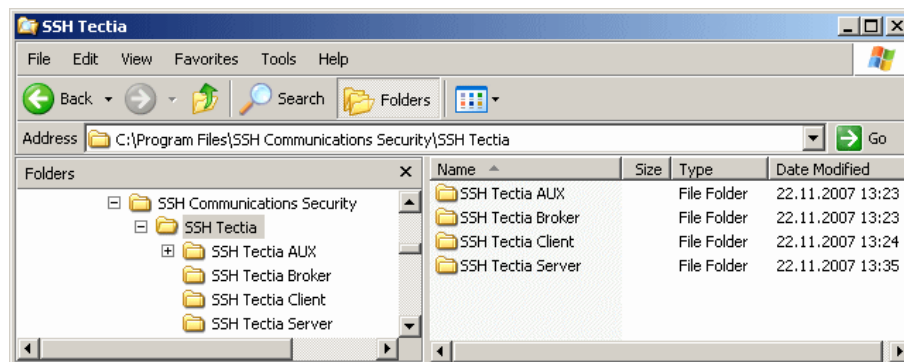


Figure 2.6. The SSH Tectia directory structure on Windows

The user-specific configurations are stored in the directories as listed below.

- `%APPDATA%\SSH\ssh-broker-config.xml`: the user-specific Connection Broker configuration file
- `%APPDATA%\SSH\global.dat`: the SSH Tectia terminal GUI configuration file
- `%APPDATA%\SSH\*.ssh2`: the SSH Tectia terminal GUI profile configuration files
- `%APPDATA%\SSH\random_seed`: the seed file for the random number generator
- `%APPDATA%\SSH\HostKeys`: the user-specific directory for known remote host keys
- `%APPDATA%\SSH\UserKeys`: the default directory for user public-key pairs
- `%APPDATA%\SSH\UserCertificates`: the default directory for user certificate key pairs
- `%APPDATA%\SSH\identification`: (*optional*) the identification file used with public-key authentication



### Note

The user-specific `%APPDATA%` directory is hidden by default. To view hidden directories, change the setting in Windows Explorer. For example, on Windows XP, select **Tools** → **Folder Options** on the menu, click the **View** tab, and select **Show hidden files and folders**.

## 2.4.3 Registry Keys on Windows

On Windows, the SSH Tectia Client installation creates the following registry keys:

- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia Broker`
- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia Broker GUI`
- `HKLM\SOFTWARE\Wow6432Node\SSH Tectia`
- `HKLM\SOFTWARE\Wow6432Node\SSH Tectia Client`
- `HKLM\SOFTWARE\Wow6432Node\SSH Tectia Connector`

## 2.5 Symlinks between ssh/scp/sftp and sshg3/scpg3/sftpg3 (on Unix)

By default, SSH Tectia Client does not create symlinks between the command-line clients `sshg3`, `scpg3` and `sftpg3`, and their earlier versions `ssh`, `scp` and `sftp`.

In case you want to make sure that the `sshg3/scpg3/sftpg3` clients are always used instead of the `ssh/scp/sftp` clients (even when the user types in `ssh/scp/sftp`) make symlinks between them by running the following script any time after the installation:

---

```
# /opt/tectia/libexec/ssh-create-4.x-compat-symlinks
```

The symlink is need as the two versions of the clients are located in different directories:

```
sshg3/scpg3/sftpg3
```

are located in `/opt/tectia/bin/sshg3`

```
ssh/scp/sftp
```

are located in `/usr/local/bin/ssh`



## Chapter 3 Getting Started with SSH Tectia Client

This chapter provides information on how to get started with SSH Tectia Client software after it has been successfully installed.

### 3.1 Product Components

SSH Tectia Client consists of the following components:

- Connection Broker: `ssh-broker-g3`, `ssh-broker-ctl`
- Secure Shell command-line tools: `sshg3`, `scp3`, `sftp3`
- Auxiliary command-line tools: `ssh-keygen-g3`, `ssh-cmpclient-g3`, `ssh-scepclient-g3`, `ssh-certview-g3`, `ssh-ekview-g3`
- SSH Tectia terminal GUI (Windows)
- SSH Tectia file transfer GUI (Windows)
- Connection Broker and SSH Tectia Configuration GUI on all supported platforms

For more information on the command-line tools, see [Appendix C](#).

### 3.2 First Login to a Remote Host

This section gives basic instructions on how you can log in from SSH Tectia Client to a Secure Shell server with the default settings. The default settings on SSH Tectia Client and SSH Tectia Server allow login with passwords, public keys, and GSSAPI.

There are separate instructions on using the SSH Tectia terminal GUI on Windows to connect to a remote server host see [Section 3.2.1](#)) and on using `sshg3` on the command line on Windows and Unix (see [Section 3.2.2](#)).

SSH Tectia Client includes a shortcut menu that helps configuring the connection settings, and in viewing the status of the connections and authentication keys. For a description of the SSH Tectia shortcut menu, see [Section A.5](#).

### 3.2.1 Logging in with SSH Tectia Terminal GUI (on Windows)

With SSH Tectia Client it is easy to establish connections to new remote host computers, and to manage the settings required for each host. The Quick Connect option allows you to quickly open new connections, minimizing the work associated with configuring each connection. It is easy to define profiles for new hosts, and save the correct settings for each.

On Windows, you can connect to a remote host by using the SSH Tectia terminal GUI as follows:

1. Open the SSH Tectia Terminal by clicking its icon on your desktop:



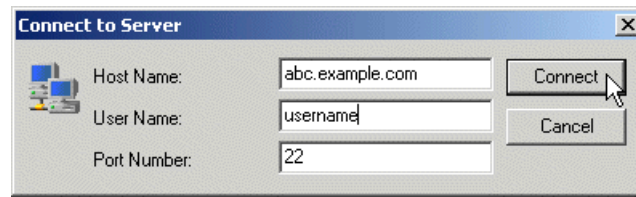
**Figure 3.1. The SSH Tectia Terminal icon**

2. SSH Tectia Terminal GUI offers several ways to open a Secure Shell connection:
  - If you already have a session open, click the **Quick Connect** command (toolbar or **File** menu). You can connect to a new remote host computer and still keep the old connection to a different host open.
  - If you have closed an earlier session, you can open a new session by hitting **Enter** or **Space** on the keyboard when the (still disconnected) terminal window is active.
    - **Enter** connects you directly to the same remote server as the previous session.
    - **Space** opens the **Connect to Server** dialog, and you can define the server where you wish to connect.
  - If you have defined connection profiles, you can also connect by clicking **File** → **Profiles** → *<Profile name>* on the menu, or clicking the **Profiles** button on the toolbar and clicking the profile name.

In this case, the settings defined in the profile (hostname, port, user name etc.) are automatically used for the connection and the **Connect to Server** dialog is not shown. For instructions on creating and editing the connection profiles, see [Section A.1.3](#).

3. This opens the **Connect to Server** dialog where you can define the host you want to connect to:





**Figure 3.2. The Connect to Server dialog**

Define the following information and click **Connect**:

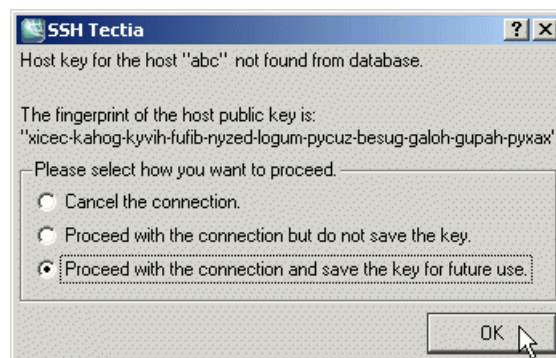
- Host Name – the FQDN, short hostname, or the IP address of the remote host
- User Name – your user name on the remote host
- Port Number – 22 is the default Secure Shell listener port.

With later sessions, the values used in the previous connection will be pre-filled.

4. The server authentication phase starts. The remote server host will provide your local computer with its host public key. The host key identifies the server host.

SSH Tectia Client checks if information on this key is already stored in your own host key directory. If not, the host key directory common to all users on your computer is checked next. If information on this host key is not found, you are asked to verify the new key.

When public-key authentication is used to authenticate the server, *the first connection is very important*. When SSH Tectia Client receives a new server host key, it will display the host identification message. For example [Figure 3.3](#):



**Figure 3.3. The host identification dialog – the first connection to a remote host**

The message displays the fingerprint of the host's public key in the SSH Babble format that is a series of pronounceable five-letter words in lower case and separated by dashes.

5. Verify the validity of the fingerprint, preferably by contacting the administrator of the remote host computer by telephone. After verifying the fingerprint, it is safe to save information on the host key for future use. You can also choose to cancel the connection, or to proceed with this connection without saving the host public key information.



## Caution

Never save a host public key without verifying its authenticity!

6. Click **OK** to close the host identification dialog.

Information on the server public key will be stored on the client-side machine so that the client can later validate the key. On SSH Tectia Client, the public key information is stored in the "%APP-DATA%\SSH\HostKeys" directory.

After the first connection, only the locally stored information about the server public key will be used in server authentication.

For more information on server authentication, see [Section 5.1](#).

7. The user authentication phase starts. You will be prompted to authenticate yourself to the server with your password or with the passphrase of your private key. The required authentication method depends on the server settings.

After the server has successfully authenticated you, the Secure Shell connection to the server is opened.

## 3.2.2 Logging in with Command-Line sshg3

You can connect to a remote host by using the `sshg3` command on the command line:

1. Enter the `sshg3` command using the following syntax:

```
$ sshg3 <hostname>
```

For example: `$ sshg3 abc.example.com.`

The basic syntax is:

```
$ sshg3 user@host#port
```

where:

- `user` - Enter a username that is valid on the remote host. The `user@` attribute is optional. If no user-name is given, the local username is assumed.
- `host` - Enter the name of the remote host as an IP address, FQDN, or short hostname. The remote host must be running a Secure Shell version 2 server.

- `port` - Enter the number of the Secure Shell listen port on the remote server. The `#port` attribute is optional. If no port is given, the default Secure Shell port 22 is assumed.

If you have defined connection profiles in the `ssh-broker-config.xml` file, you can also connect by using the name of the connection profile, for example:

```
$ sshg3 profile1
```

In this case, the settings defined in the profile (hostname, port, username etc.) are used for the connection. For instructions on creating and editing the connection profiles, see [the section called “The profiles Element”](#).

For more information on the command-line commands and options, see [Appendix C](#).

2. The server authentication phase starts. The server sends its public key to the client for validation (when server public-key authentication is used).

SSH Tectia Client checks if this key is already stored in your own host key directory. If not, the host key directory common to all users on your computer is checked next.

If the host key is not found, you are asked to verify it.

When SSH Tectia Client receives a new host public key, a host identification message is displayed. For example:

```
$ sshg3 user@host
Host key not found from database.
Key fingerprint:
xecic-fifub-kivyh-kohag-zedyn-logum-pycuz-besug-galoh-gupah-xaxby
You can get a public key's fingerprint by running
% ssh-keygen-g3 -F publickey.pub
on the keyfile.
Are you sure you want to continue connecting (yes/no)?
```

The message shows the fingerprint of the host's public key in the SSH Babble format that is a series of pronounceable five-letter words in lower case and separated by dashes.

3. Verify the validity of the fingerprint, preferably by contacting the administrator of the remote host computer by telephone.

After the fingerprint has been verified and found to be correct, it is safe to save the key and continue connecting. You can also select to cancel the connection, or to proceed with the connecting without saving the key.

If you choose to save the server public key, relevant information about the key will be stored on the client host in directory `$HOME/.ssh2/hostkeys` on Unix or in `%APPDATA%\SSH\HostKeys` on Windows. After the first connection, the locally stored information about the server public key will be used in server authentication.

For more information on server authentication, see [Section 5.1](#).

4. The user authentication phase starts. You will be prompted to authenticate yourself to the server with your password or with the passphrase of your private key (if your public key has already been uploaded to the server). The required authentication method depends on the server settings.

After the server has successfully authenticated you, the Secure Shell connection to the server is opened.

### 3.3 Using Public-Key Authentication

Public-key authentication is based on the use of digital signatures. To use public-key authentication, you must first create a key pair on the client, and upload the public key to the server. For instructions, see [Section 5.4](#).

At connection establishing phase, the server sends SSH Tectia Client a challenge. Sign the challenge with the passphrase of your private key. After the server has successfully completed user authentication, the Secure Shell connection to the server is opened.


The Connection Broker operates automatically as an authentication agent. It offers an easy method for utilizing also digital certificates and smart cards. The authentication forwarding functionality allows the forwarding of public-key authentication over several Secure Shell connections. The Connection Broker is started automatically when you start SSH Tectia Client.

## Chapter 4 Configuring SSH Tectia Client

SSH Tectia Client includes two configuration tools:

- **SSH Tectia Configuration** tool (behind the  icon) is used to edit the connection and authentication settings.

A component called Connection Broker handles all cryptographic operations and authentication-related tasks for SSH Tectia Client, so all authentication and connection profile settings are made in the Connection Broker configuration.

- **User Interface Settings** tool (behind the  icon) is used to edit the SSH Tectia terminal and file transfer GUI.

For instructions on configuring the user interface, see [Appendix B](#).

This chapter explains the basic idea about the Connection Broker configuration.

For a detailed description of the configuration options related to Connection Broker, see [Appendix A](#).

### 4.1 Configuration Files

The following files, located in the `/etc/ssh2` directory, are used to store the SSH Tectia Client configuration information:

- `/etc/ssh2/ssh-broker-config.xml`: the global Connection Broker configuration file
- `/etc/ssh2/hostkeys`: the global directory for known remote server host keys

The user-specific configurations are stored in each user's `$HOME/.ssh2` directory:

- `$HOME/.ssh2/ssh-broker-config.xml`: the user-specific Connection Broker configuration file
- `$HOME/.ssh2/hostkeys`: the user-specific directory for known remote server host keys
- `$HOME/.ssh2/identification`: the identification file used with public-key authentication

- `$HOME/.ssh2/random_seed`: the random number seed file for cryptographic operations

### 4.1.1 Editing the Configuration Files

SSH Tectia Client includes a default configuration that can get you started, but to tailor the SSH Tectia Client behaviour according to the needs of your environment, you can edit the existing configuration and add settings to the configuration files.

You can edit the `ssh-broker-config.xml` file with your favorite XML or text editor. When you make modifications to the Connection Broker configuration file, a user-specific copy of the file will be stored to `$HOME/.ssh2` which is the default location for the user-specific configuration files.

One of the first items to configure are the user and server authentication settings, for example creating public keys for the users and uploading them to remote servers. For instructions on defining the authentication settings, see [Chapter 5](#), and for the authentication-related options in the configuration file, see [authentication-methods](#).

Another immediately useful item to configure are connection profiles that aid in connecting repeatedly to the same remote servers. For instructions on adding connection profiles in the configuration file, see [the section called “The profiles Element”](#), or to create the profiles via the GUI, see [Section A.1.3](#).

The Connection Broker configuration file `ssh-broker-config.xml` must be a valid XML file. For details about the Connection Broker configuration options, see [ssh-broker-config\(5\)](#).

## 4.2 Command-Line Options

In addition to the configuration file and environment variables, also command-line options can be used to configure the SSH Tectia Client functions. The command-line options override values specified in environment variables and the configuration file.

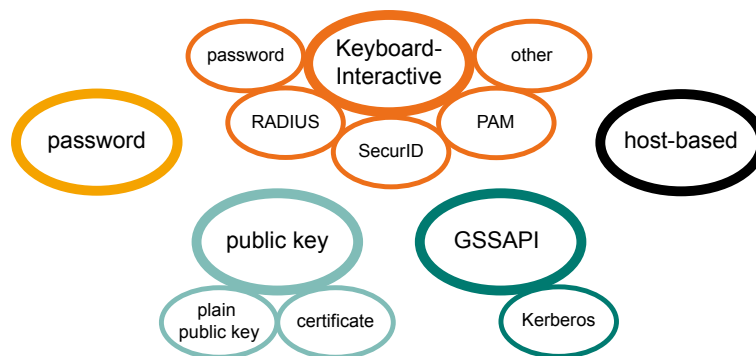
For a list of the available options, see the man pages of the command-line tools [sshg3\(1\)](#), [scp3\(1\)](#), and [sftp3\(1\)](#).

## Chapter 5 Authentication

The Secure Shell protocol used by the SSH Tectia client/server solution provides mutual authentication – the client authenticates the server and the server authenticates the client user. Both parties are assured of the identity of the other party.

The remote Secure Shell server host can authenticate itself using either traditional public-key authentication or certificate authentication.

Different methods can be used to authenticate Secure Shell client users. These authentication methods can be combined or used separately, depending on the level of functionality and security you want.



**Figure 5.1. User authentication methods**

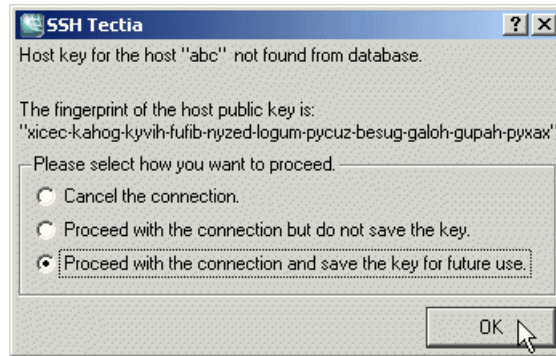
User authentication methods used by SSH Tectia Client by default are: public-key, password, keyboard-interactive, and GSSAPI authentication. Public-key and certificate authentication are combined into the public-key authentication method.

For reference information on authentication methods, see technical note *SSH Tectia Client/Server User Authentication Methods* at <http://www.ssh.com/resources/>.

## 5.1 Server Authentication with Public Keys

The server is authenticated with a digital signature based on a DSA or RSA public-key algorithm. At the beginning of the connection, the server sends its public key to the client for validation.

Server authentication is done during Diffie-Hellman key exchange through a single public-key operation. When public-key authentication is used to authenticate the server, the first connection is very important. During the first connection the client will display a message similar to the one in [Figure 5.2](#).



**Figure 5.2. SSH Tectia Client on Windows – first connection to a remote host**



### Caution

Never save a host public key without verifying its authenticity!

To help you to verify the identity of the server host, the message displays a fingerprint of the host's public key. The fingerprint is represented using the SSH Babble format, and it consists of a series of pronounceable five-letter words in lower case and separated by dashes.

Verify the validity of the fingerprint, for example by contacting the administrator of the remote host computer (preferably by telephone) and asking the administrator to verify that the key fingerprint is correct. If the fingerprint is not verified, it is possible that the server you are connecting to is not the intended one (this is known as a *man-in-the-middle attack*).

After verifying the fingerprint, it is safe to continue connecting. Relevant information about the server public key will then be stored on the client-side machine. On SSH Tectia Client on Unix it is stored in the `$HOME/.ssh2/hostkeys` directory. On SSH Tectia Client on Windows it is stored in the `%APP-DATA%\SSH\HostKeys` directory.

The stored information on the host keys is used in subsequent connections to those remote hosts. SSH Tectia Client checks which type of a host key (RSA or DSA) it possesses for a particular server, and automatically chooses the key exchange algorithm to be used in the connection between the client and server accordingly. This makes it quicker to connect to hosts for which only one type of host key has been stored.



When the `strict-host-key-check` is disabled (as by default), SSH Tectia Client will log information about changed and new host public keys with their fingerprints in the syslog (on Unix) or Event Viewer (on Windows).

## 5.1.1 Host Key Storage Formats

When the host key is received during the first connection to a remote host (or when the host key has changed) and you choose to save the key, its filename is stored in hashed format, `keys_hhh...`, where `hhh` is a hash of the host port and name. The saved file contains a hash of the host's public key. A salt is included in the hash calculations. The value of the salt is stored in the file `salt` in the same directory as the host keys (`$HOME/.ssh2/hostkeys` on Unix, `%APPDATA%\SSH\HostKeys` on Windows). The hashed host key format is a security feature to make address harvesting on the hosts difficult.

In the plain (traditional) format, the name of a host key file includes the host's name and port, as in `key_22_host.example.com.pub`, and the file contains the host's public key in plaintext format.

The storage format can be controlled with the `filename-format` attribute of the `known-hosts` element of the `ssh-broker-config.xml` configuration file. The attribute value must be `plain` or `hash` (default). See [known-hosts](#) for details.

```
<known-hosts path="$HOME/.ssh2/hostkeys" filename-format="plain" />
```

If you are adding the keys manually, the keys should be named with the `key_<port>_<host>.pub` pattern, where `<port>` is the port the Secure Shell server is running on and `<host>` is the hostname you use when connecting to the server (for example, `key_22_alpha.example.com.pub`).

If both the hashed and plaintext format keys exist, the hashed format takes precedence.

Note that the host identification is different based on the host name and port the client is connecting to. The hostname can occur in 3 different formats: fully qualified domain name (FQDN), short hostname, or IP address. The host key for each name format has to be saved separately, as they are not mutually exchangeable.

The host key is saved under the hostname format used in the login. For example, if you want to use all the hostname formats when connecting to a remote host named `alpha`, connect to the host first with the following commands and save the host key under all three names:

- `sshg3 user@alpha`

produces the key with the short hostname (in plain format `key_22_alpha.pub`)

- `sshg3 user@alpha.example.com`

produces the key with FQDN (in plain format `key_22_alpha.example.com.pub`)

- `sshg3 user@10.1.101.10`

produces the key with IP-address (in plain format `key_22_10.1.101.10.pub`)

Also if you need to connect to the same host but different port, your client needs a separate host key for that purpose; for example `key_22_alpha.pub` and `key_222_alpha.example.com.pub`.

After the first connection, the locally stored information about the server public key will be used in server authentication.

## 5.1.2 Using the System-Wide Host Key Storage

If a host key is not found in the user-specific host key directory, it is next searched on Unix from the `/etc/ssh2/hostkeys` directory and on Windows from the "%ALLUSERSPROFILE%\Application Data\SSH\HostKeys" directory. Host key files are not automatically put in the system-wide directory but they have to be updated manually by the system administrator (`root`) or by using SSH Tectia Manager.

If SSH Tectia Manager is not used for distributing the host keys, you can follow the instructions below for doing it manually. The instructions reflect the Unix file paths but are applicable also to Windows. Simply replace the Unix paths with the corresponding Windows paths.

### 5.1.2.1 Storing Keys in the Hashed Format

To obtain and store hashed remote host keys in the system-wide storage:

1. Select a client-side user whose `$HOME/.ssh2/hostkeys` will be the basis for the system-wide `/etc/ssh2/hostkeys`. The user should have administrative privileges, as placing the keys to the system-wide location requires them.

The same user account must also be used to maintain the system-wide `/etc/ssh2/hostkeys` later on if the host key on some server changes. The process is to maintain the user's host keys in the `$HOME/.ssh2/hostkeys` directory and then replicate the changes to the system-wide `/etc/ssh2/hostkeys` directory.

2. Make sure that the `$HOME/.ssh2/hostkeys` directory is empty when obtaining the keys for the first time, or that the saved host keys are intentional.

If you need to obtain new keys later, the same `$HOME/.ssh2/hostkeys/salt` file has to be used.

3. Connect with SSH Tectia Client to the remote server, verify the fingerprint, and save the key.

Repeat this step as many times as there are remote servers. Note that you do not have to complete the user authentication, only the key exchange part of the Secure Shell connection.

4. Once you have obtained all the host keys you wish to maintain in the system-wide location, place the keys to the system-wide location, for example by running the following commands:

```
# mkdir /etc/ssh2/hostkeys
# cp -p $HOME/.ssh2/hostkeys/* /etc/ssh2/hostkeys
```

Note that also the salt file (`$HOME/.ssh2/hostkeys/salt`) has to be copied so that SSH Tectia Client is able to identify the hashed host keys. Also if multiple users contribute to the system-wide `/etc/ssh2/hostkeys` directory, they have to share the same salt file.

After creating the system-wide location for host keys, you can maintain it by using the **ssh-keygen-g3** tool.

The following copy examples show the most frequently needed commands for host key storage maintenance. The commands use the user-specific hostkey storages (`$HOME/.ssh2/hostkeys` and possibly the `$HOME/.ssh/known_hosts` file) as the source. If keys are to be copied from a different source, you need to append an appropriate `--hostkeys-directory` or `--hostkey-file` option to the command.

To copy the key of a new host called 'alpha' from the user-specific hostkey storage to the system-wide directory, enter command:

```
# ssh-keygen-g3 --append=no --overwrite=no \
--copy-host-id alpha /etc/ssh2/hostkeys
```

In this case, if a key for server 'alpha' already exists, the command will fail and the key will not be updated.

To add additional keys to a known host, enter command:

```
# ssh-keygen-g3 --append=yes --copy-host-id alpha /etc/ssh2/hostkeys
```

To update the key of a known host, enter command:

```
# ssh-keygen-g3 --append=no --copy-host-id alpha /etc/ssh2/hostkeys
```

To remove a host from the known hosts list, enter command:

```
# ssh-keygen-g3 --hostkeys-directory /etc/ssh2/hostkeys \
--delete-host-id alpha
```

For more detailed information on the **ssh-keygen-g3** tool, see [ssh-keygen-g3\(1\)](#).

### 5.1.2.2 Storing Keys in the Plain Format

To obtain and store traditional remote host keys in the system-wide storage:

1. As a server-side user, copy the `/etc/ssh2/hostkey.pub` file from the server as `key_<port>_<host-name>.pub` to the `/etc/ssh2/hostkeys/` directory on the client.

You can do this as a non-privileged user on the server but you must be privileged user, for example `root`, on the client.

2. Use secure means to transfer the file or verify the fingerprint matches after the transfer with the **ssh-keygen-g3** option `-F`, for example on SSH Tectia Server on Unix:

```
$ ssh-keygen-g3 -F /etc/ssh2/hostkey.pub
```

On the client:

```
# ssh-keygen-g3 -F /etc/ssh2/hostkeys/key_<port>_<hostname>.pub
```

Note that the identification is different based on the host and port the client is connecting to. Also connection with IP is considered a different host as well as connection to same host but different port. You can copy the same traditional key\_<port>\_<hostname>.pub to all these different names.

### 5.1.3 Resolving Hashed Host Keys

SSH Tectia Client includes a tool to resolve which hashed host key belongs to which server. As there can be several server host keys stored on the client-side host, and the file name does not show the server name in, it is sometimes necessary to check if a certain server public key is stored on the client host.

The command syntax is:

```
ssh-keygen-g3 -F <servername>@<port>
```

For example:

```
ssh-keygen-g3 -F server1@222
```

The tool shows the location and the fingerprint of the requested server's public key or keys (the fingerprint in the SSH babble format). For example:

```
Fingerprints for key 'server1#222':
  (from location
    /etc/ssh/ssh_known_hosts:1 ("server1 ssh-dss AAAAB3...")
    (publickey-knownhosts))
xical-dohoz-fafur-ciper-vucam-munod-rykic-nabiv-nigag-fatif-pixex
  (from location
    /home/user44/.ssh/known_hosts:2 ("|1|84+eBlqwbSSvSe0GY...")
    (publickey-knownhosts))
xuvob-vodyt-dilib-koryc-cadek-ryfuv-mufut-bupyb-resuz-fadyz-taxox
```

The port definition is optional in the command. If no port is given, the default Secure Shell port 22 is assumed. For example:

```
ssh-keygen-g3 -F server2
Fingerprint for key 'server2':
  (from location
    /home/user44/.ssh2/hostkeys/keys_bf53882dc47bb767edf161a4f636917f8358d635
    (publickey-file))
xuvin-zitil-ducid-gevil-vysok-buviz-nynun-pinat-tylev-gusez-dyxix
```

If no keys are found for the given server, the ssh-keygen-g3 -F command will report where it looked for the keys, and will conclude as follows:

```
/ No keys found from any key directories or known_hosts files.
```

## 5.1.4 Using the OpenSSH known\_hosts File

SSH Tectia Client supports also the OpenSSH-style known\_hosts file that contains the public key data of known server hosts, and reads the file by default from the default location, from the user-specific file `$HOME/.ssh/known_hosts` or from the system-wide file `/etc/ssh/ssh_known_hosts`. Both hashed and plain-format host keys are supported.

In case you wish to define other files to be used for the known host keys, you can specify the files in the Connection Broker configuration file `ssh-broker-config.xml` by using the `known-hosts` element. Several file locations can be defined to be checked for known host keys, and the Connection Broker will read them in the order they are defined in the `ssh-broker-config.xml` file. Since the configuration file settings will override the default behaviour, you need to define also the default locations of the OpenSSH-style known\_hosts file, in case you want them all to be read. For example:

```
<general>
...
<known-hosts path="/home/username/.ssh/known_hosts" />
<known-hosts path="/etc/ssh/ssh_known_hosts" />
<known-hosts path="/home/.ssh2/hostkeys" />
<known-hosts path="/u/username/.ssh2/hostkeys" />
</general>
```

You can disable OpenSSH known\_hosts file handling by defining an empty setting: `known-hosts path=""`. After this, only the SSH Tectia-related hostkey directories will be used.

The OpenSSH known\_hosts file is never automatically updated by SSH Tectia Client. New host keys are always stored in the SSH Tectia `$HOME/.ssh2/hostkeys` directory or in the directory configured as the last one in `ssh-broker-config.xml`. See [known-hosts](#) for details.

## 5.2 Server Authentication with Certificates

Server authentication with certificates happens similarly to server authentication with public keys, except that the possibility of a man-in-the-middle attack during the first connection to a particular server is eliminated. The signature of a certification authority in the server certificate guarantees the authenticity of the server certificate even in the first connection.

A short outline of the server authentication process with certificates is detailed below:

1. The server sends its certificate (which contains a public key) to the client. The packet also contains random data unique to the session, signed by the server's private key.
2. As the server certificate is signed with the private key of a certification authority (CA), the client can verify the validity of the server certificate by using the CA certificate.
3. The client checks that the certificate matches the name or the IP address of the server. When `end-point-identity-check` is enabled in the Connection Broker configuration, the client compares the server's

hostname or IP address to the Subject Name or Subject Alternative Name (DNS Address) specified in the server certificate.

If `end-point-identity-check` is disabled in the Connection Broker configuration, the fields in the server host certificate are not verified and the certificate is accepted based on the validity period and CRL check only.



## Caution

Disabling the endpoint identity check on the client is a security risk. Then anyone with a certificate issued by the same trusted CA that issues the server host certificates can perform a man-in-the-middle attack on the server.

4. The client verifies that the server has a valid private key by checking the signature in the initial packet.

During authentication the system checks that the certificate has not been revoked. This can be done either by using the Online Certificate Status Protocol (OCSP) or a certificate revocation list (CRL), which can be published either in an LDAP or HTTP repository.

OCSP is automatically used if the certificate contains a valid **Authority Info Access** extension, or an OCSP responder has been separately configured. If no OCSP responder is defined or the OCSP connection fails, CRLs are used. If LDAP is used as the CRL publishing method, the LDAP repository location can also be defined in the `ssh-broker-config.xml` file.

### 5.2.1 Using the Configuration File (Unix)

When configuring the client, it must be set up to trust the CA certificate and to access the certificate revocation list (CRL).

To configure the client to trust the server's certificate, perform the following tasks:

1. Copy the CA certificate(s) to the client machine. You can either copy the X.509 certificate(s) as such, or you can copy a PKCS #7 package including the CA certificate(s).

Certificates can be extracted from a PKCS #7 package by specifying the `-7` flag with **ssh-keygen-g3**.

2. Define the CA certificate(s) to be used in host authentication in the `ssh-broker-config.xml` file under the `general` element:

```
<cert-validation end-point-identity-check="yes"
    http-proxy-url="http://proxy.example.com:800">
  <ldap-server address="ldap://ldap.example.com:389" />
  <ocsp-responder url="http://ocsp.example.com:8090" validity-period="0" />
  <dod-pki enable="no" />
  <ca-certificate name="ssh_cal"
    file="ssh_cal.crt"
    disable-crls="no"
```

```
use-expired-crls="100" />
</cert-validation>
```

The client will only accept certificates issued by the defined CA(s).

You can disable the use of CRLs by setting the `disable-crls` attribute of the `ca-certificate` element to "yes".



### Note

CRL usage should only be disabled for testing purposes. Otherwise it is highly recommended to always use CRLs.

Also define the LDAP server(s) or OCSP responder(s) used for CRL checks. Defining the LDAP server is not necessary if the CA certificate contains a CRL distribution point extension.

3. If the CA services (OCSP, CRL) are located behind a firewall, define also the SOCKS server in the `ssh-broker-config.xml` file. The SOCKS server is defined inside `cert-validation` with the `socks-server-url` element.

## 5.2.2 Using the GUI

Using the SSH Tectia Configuration tool to manage CA certificates is described in [Section A.1.5.2](#).

## 5.3 User Authentication with Passwords

The password authentication method is the easiest to implement, as it is set up by default. Since all communication is encrypted, passwords are not available for eavesdroppers.

On a Unix system, password authentication uses the `/etc/passwd` or `/etc/shadow` file, depending on how the passwords are set up. The shadow password files can be used on Linux and Solaris servers, but not on HP-UX or AIX servers.

On Windows, password authentication uses the Windows password to authenticate the user at login time.

### 5.3.1 Using the Configuration File (Unix)

To enable password authentication on the client, the `authentication-methods` element of the `ssh-broker-config.xml` file must contain an `authentication-method` element with the `name` attribute value `password`:

```
<authentication-methods>
...
  <authentication-method name="password" />
</authentication-methods>
```

Other authentication methods can be listed in the configuration file as well. Place the least interactive method first (password is usually the last one).

### 5.3.2 Using Stored Passwords in Connection Profiles

In connection profiles that will be used in non-interactive connections, it is also possible to use passwords stored to the SSH Tectia Client configuration or to the system.

In the Connection Broker configuration file `ssh-broker-config.xml`, the stored passwords are configured with the element `password`, with the following syntax:

```
<profiles>
  <profile>
    <authentication-method name="password" />
    <password file="path/to/file" />
    <password command="path/to/script_or_program" />
    ...
  </profile>
  ...
</profiles>
```

The `password` element can be used to specify a user password that the client will send as a response to password authentication.

The password can be given directly in the `string` attribute, but safer alternatives are to define either a path to a file containing the password in the `file` attribute, or to use the `command` attribute to define a path to a program or script that outputs the password.

When using the `command` attribute to refer to a shell script, make sure the script also defines the user's shell, and outputs the actual password. Otherwise the executed program fails, because it does not know what shell to use for the shell script. For example, if the password string is defined in a file named `my_password.txt`, and you want to use the bash shell, include these lines in the script:

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```



#### Caution

If the password is given using this option, it is extremely important that the `ssh-broker-config.xml` file, the password file, or the program are not accessible by anyone else than the intended user.



#### Note

Any password given with the command-line options will override this setting.

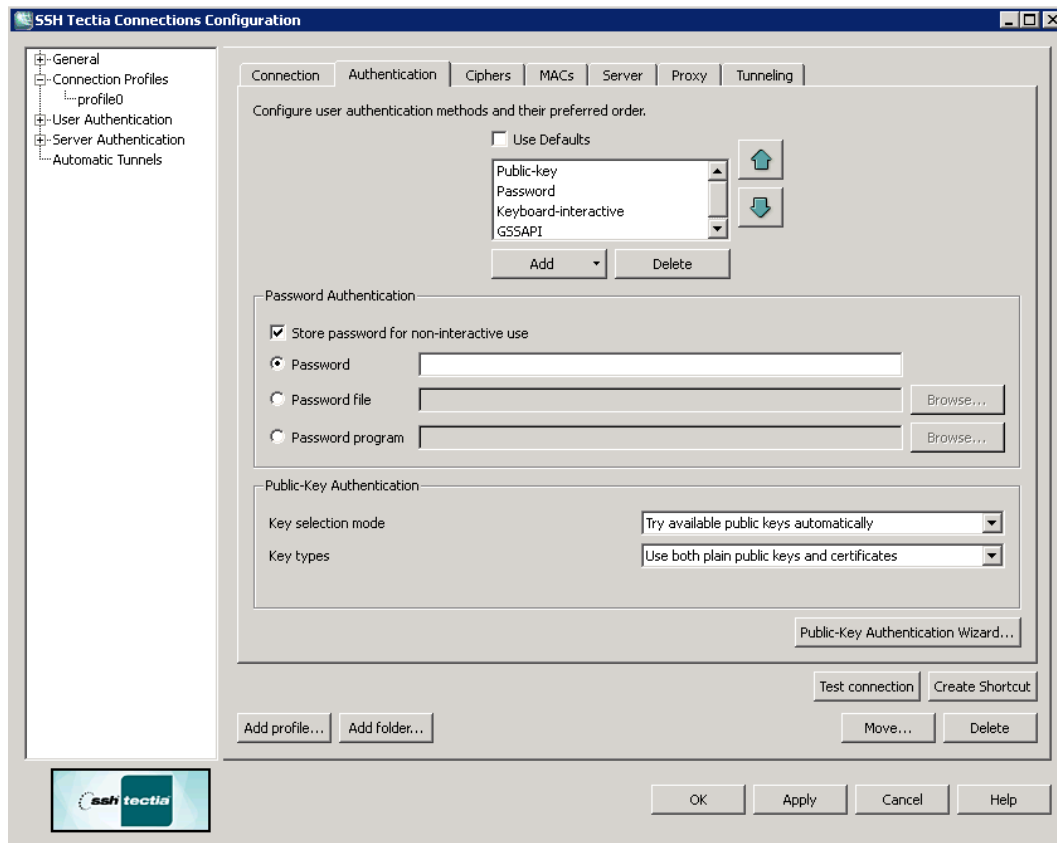


Via the SSH Tectia Configuration GUI, the stored passwords are configured on the **Connection profiles** → **Authentication** tab. Select **Store password for non-interactive use** and define the password or the path to the password file or program.



## Caution

If you choose to use stored passwords, it is extremely important that the SSH Tectia Client host and the password file or program are not accessible by anyone else than the intended user.



**Figure 5.3. Configuring authentication methods for the profile**

To store the password as such in the configuration, enter the password directly in the **Password** field.

To use a file containing the password, select **Password file** and enter the path to the file in the field.

To use a program or a script that outputs the password, select **Password program** and enter the path to the program in the field.



## Note

The user is required to have adequate permissions to the password file and to the password program. The file or the program executable must be owned by the user, local administrator or a member in the local admin group, and the file must have the `allow-type` permissions for administrators.

### 5.3.3 Using the GUI

Using the SSH Tectia Configuration tool to manage authentication methods is described in [Section A.1.3.2](#).

## 5.4 User Authentication with Public Keys

Public-key authentication is based on the use of digital signatures. Each user creates a pair of key files. One of these key files is the user's public key, and the other is the user's private key. The server knows the user's public key, and only the user has the private key.

The key files must be stored to a location where the user has the `write` rights, (and `read` rights), but that is not accessible to others. These user-specific rights are required for the `key.pub` file, the `authorized_keys` directory, and to the `authorization` file, if used.

When the user tries to authenticate, the client sends a signature to the server, and the server checks for matching public keys. If the key is protected with a passphrase, the server requests the user to enter the passphrase.

Remember that your private-key file is used to authenticate you. Keep your private-key file in a secure place and make sure that no one else has access to it. If anyone else can access your private-key file, they can attempt to log in to the remote host computer pretending to be you. Define a passphrase to protect your private key, whenever possible. On a machine shared by several users, make sure that the permission settings do not allow others to access your private key.



## Caution

Do not store your private keys in a location accessible to other users.

Also note that if you are using the Windows roaming profiles functionality, your personal settings will be replicated with the roaming profile server. If you store your private keys in the default location (under the profile folder of your Windows user account) your private keys may be susceptible to a malicious user listening to the network traffic. Therefore the User Settings folder should not be a directory that is used in profile roaming.

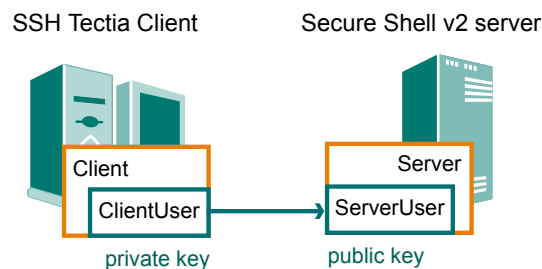
To use public-key authentication with SSH Tectia Client, do the following actions:

1. Generate a key pair. You can generate your own key files with the help of a built-in Public-Key Authentication Wizard on Windows (see [Section 5.4.3](#)), or with **ssh-keygen-g3** on Unix or Windows command line (see [Section 5.4.1](#)).

You can also import existing keys on the **Keys and Certificates** page of the SSH Tectia Configuration tool. See [Section A.1.4.1](#).

2. Upload your public key to the remote host computer. On Windows, you can do this automatically (see [Section 5.4.3.2](#)). On Unix and Windows, you can also copy the public key manually (see [Section 5.4.2](#)).

In the following instructions, *Server* is the remote host running the Secure Shell server that you are trying to connect to. *ServerUser* is the username on *Server* that you are logging in as. *Client* is the host running the Secure Shell client (SSH Tectia Client). *ClientUser* is the username on *Client* that should be allowed to log in to *Server* as *ServerUser*. See [Figure 5.4](#).



**Figure 5.4. User public-key authentication**

The instructions assume that *ClientUser* is allowed to log in to *Server* as *ServerUser* using some other authentication method (usually password).

## 5.4.1 Creating Keys with ssh-keygen-g3

To create a public key pair, run **ssh-keygen-g3** on *Client*:

```
$ ssh-keygen-g3
Generating 2048-bit dsa key pair
 9 oOo.oOo.oOo
Key generated.
2048-bit dsa, ClientUser@Client, Thu Jan 22 2008 12:09:46 +0200
Passphrase :
Again :
Private key saved to /home/ClientUser/.ssh2/id_dsa_2048_a
Public key saved to /home/ClientUser/.ssh2/id_dsa_2048_a.pub
```

When run without options, **ssh-keygen-g3** asks for a passphrase for the new key. Enter a sufficiently long (20 characters or so) sequence of any characters (spaces are OK).

The new authentication key pair consists of two separate files. One of the keys is your private key which must *never* be made available to anyone but yourself. The private key can only be used together with the passphrase.

On Unix, the key pair is by default stored in your `$HOME/.ssh2` directory (created by **ssh-keygen-g3** if it does not exist previously). On Windows, the key pair is by default stored in your `%APPDATA%\SSH\UserKeys` directory.

In the example above, the private key file is `id_dsa_2048_a`. The public key file is `id_dsa_2048_a.pub`, and it can be distributed to other computers.

By default, **ssh-keygen-g3** creates a 2048-bit DSA key pair. RSA keys can be generated by specifying the `-t` option with **ssh-keygen-g3**. Key length can be specified with the `-b` option. For automated jobs, the key can be generated without a passphrase with the `-P` option, for example:

```
$ ssh-keygen-g3 -t rsa -b 1536 -P
```

For more information on the **ssh-keygen-g3** options, see [ssh-keygen-g3\(1\)](#).

## 5.4.2 Uploading Public Keys Manually

All commands in this section are shown using **sshg3** and **scpg3** from the machine running SSH Tectia Client. Server-side configuration can also be done by logging in to the remote server and entering the commands locally.

To enable public-key authentication with your key pair:

1. Place your keys in a directory where the Connection Broker can locate them.

By default, the Connection Broker attempts to use each key found in the `$HOME/.ssh2` directory on Unix, or in the `%APPDATA%\SSH\UserKeys` and `%APPDATA%\SSH\UserCertificates` directories on Windows.

You can also add other directory locations for keys on the **Keys and Certificates** page of the SSH Tectia Configuration tool. See [Section A.1.4.1](#). On Unix, you can use the `general/key-stores/key-store` element in the `ssh-broker-config.xml` file. See [the section called “Key Store Configuration Examples”](#).

2. (Optional) Create an identification file.

Using the `identification` file is not necessary if all your keys are stored in the default directory and you allow all of them to be used for public-key and/or certificate authentication. If the `identification` file does not exist, the Connection Broker attempts to use each key found in the default directory. If the `identification` file exists, the keys listed in it are attempted first.

Create a file called `identification`, on Unix in your `$HOME/.ssh2` directory, or on Windows in your `%APPDATA%\SSH` directory.

Edit it with your favorite text editor to include the following line (replace `id_dsa_2048_a` with the filename of the private key):

```
IdKey      id_dsa_2048_a
```

The keys are assumed to be in the same directory with the `identification` file, but also an absolute or a relative path can be given. For example, on Windows:

```
IdKey      UserKeys\id_dsa_2048_a
```

The identification file can contain several key IDs. For more information on the syntax of the identification file, see [\\$HOME/.ssh2/identification](#).

3. Connect to Server using some other authentication method and create a `.ssh2` (and `.ssh2/authorized_keys`), or a `.ssh` directory under your home directory if it does not exist already.

Depending on the server version the remote host is running, run one of the following commands:

- SSH Tectia Server on Unix or z/OS:

```
$ sshg3 ServerUser@tectia_server mkdir .ssh2
```

If you do not want to use an authorization file on SSH Tectia Server 5.x or later on Unix, create also the `authorized_keys` directory:

```
$ sshg3 ServerUser@tectia_unix mkdir .ssh2/authorized_keys
```

- SSH Tectia Server on Windows:

```
$ sshg3 ServerUser@tectia_win "cmd /c mkdir .ssh2"
```

If you do not want to use an authorization file on SSH Tectia Server 5.x or later on Windows, create also the `authorized_keys` directory:

```
$ sshg3 ServerUser@tectia_win mkdir "cmd /c mkdir .ssh2/authorized_keys"
```

- OpenSSH server on Unix or z/OS:

```
$ sshg3 ServerUser@open_server mkdir .ssh
```

4. Copy the public key to Server.

Depending on the server version the remote host is running, do one of the following actions:

- SSH Tectia Server 5.x or later on Unix and Windows:

Use SCP to upload your public key to the server, to your `authorized_keys` directory (by default `$HOME/.ssh2/authorized_keys` on Unix servers, or `%USERPROFILE%\.ssh2\authorized_keys` on Windows servers):

```
$ scp3 id_dsa_2048_a.pub ServerUser@tectia_server:~/.ssh2/authorized_keys/
```

- SSH Tectia Server 4.x on Unix and Windows:

Use SCP to upload your public key to the server (by default to the `$HOME/.ssh2` directory on Unix and to the `%USERPROFILE%\ssh2` directory on Windows servers):

```
$ scp3 id_dsa_2048_a.pub ServerUser@tectia4x_server:.ssh2/
```

- SSH Tectia Server for IBM z/OS:

The public key must be converted to the EBCDIC format. This can be done by including the `dst-site` command-line options (when using `scp3`), or the `site` commands (when using `sftp3`) in the file transfer command.

Use SCP to upload your public key to the server (by default to the `$HOME/.ssh2` directory):

```
$ scp3 --dst-site="C=ISO8859-1,D=IBM-1047,X=TEXT" id_dsa_2048_a.pub \
ServerUser@tectia_zos:$HOME/.ssh2/
```

- OpenSSH server on Unix:

Use SCP to upload your public key to the server, to your `$HOME/.ssh` directory:

```
$ scp3 id_dsa_2048_a.pub ServerUser@open_unix:.ssh/
```

## 5. Create an authorization or `authorized_keys` file on Server.

Depending on the server version the remote host is running, do one of the following actions:

- SSH Tectia Server 5.x or later on Unix and Windows do not require an authorization file if the public keys are stored in the user's `authorized_keys` directory. However, an authorization file may be optionally used. See instructions for creating the file below in the SSH Tectia Server 4.x information.
- SSH Tectia Server 4.x on Unix and Windows and SSH Tectia Server for IBM z/OS require an `authorization` file stored in the user's `.ssh2` directory. The authorization file specifies the public keys that are authorized for login.

Add the key entry to the authorization file. On a Unix or z/OS server:

```
$ sshg3 ServerUser@tectia_server "echo Key id_dsa_2048_a.pub >> \
.ssh2/authorization"
```

On a Windows server:

```
$ sshg3 ServerUser@tectia4x_win "cmd /c echo Key id_dsa_2048_a.pub >> \
.ssh2\authorization"
```

An example authorization file is shown below (by default `$HOME/.ssh2/authorization` on Unix and z/OS servers, and `%USERPROFILE%\ssh2\authorization` on Windows servers):

```
Key      id_dsa_2048_a.pub
```

This directs SSH Tectia Server to use `id_dsa_2048_a.pub` as a valid public key when authorizing your login.

- OpenSSH server requires that the public key is converted to the OpenSSH public-key file format and stored in the `authorized_keys` file in the user's `.ssh` directory.

Convert the public key to the OpenSSH public key file format on the server and append it to your `$HOME/.ssh/authorized_keys` file. This can be done with a remote command on an OpenSSH server as follows:

```
$ sshg3 ServerUser@open_server "ssh-keygen -i -f id_dsa_2048_a.pub >> \
.ssh/authorized_keys"
```

6. Make sure that public-key authentication is enabled in the `ssh-broker-config.xml` file (it is enabled by default).

```
<authentication-methods>
  <auth-publickey />
  ...
</authentication-methods>
```

Other authentication methods can be listed in the configuration file as well. Place the least interactive method first.

Assuming Server is configured to allow public-key authentication to your account, you should now be able to log in from Client to Server using public-key authentication.

Try to log in:

```
Client$ sshg3 Server
```

You should be prompted for the passphrase of the private key. After you have entered the passphrase, a Secure Shell connection will be established.

### 5.4.3 Creating Keys with the Public-Key Authentication Wizard (Windows)

On Windows, you can use the SSH Tectia **Public-Key Authentication Wizard** to generate a key pair. The wizard will generate two key files, your private key and your public key.

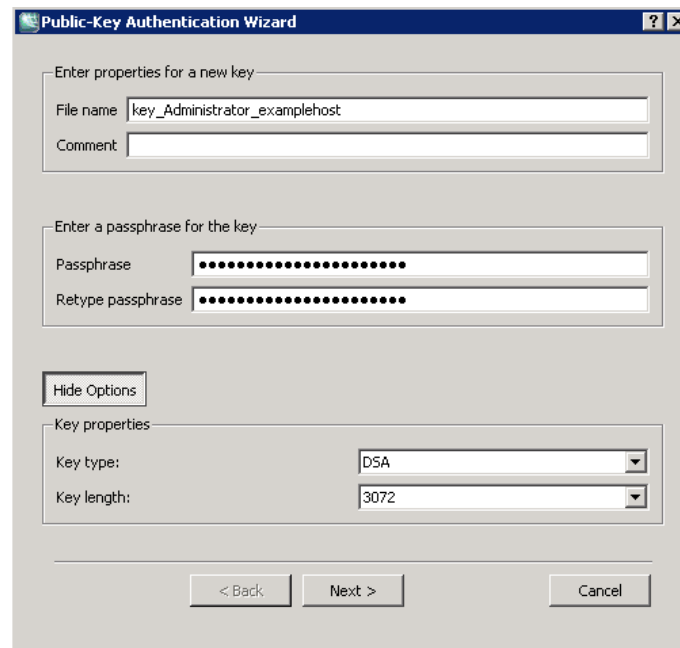
The new private and public key will be stored on your local computer in the `%APPDATA%\SSH\UserKeys` directory. The private key file has no file extension, and the public key has the same base file name as the private key, but with `.pub` as the file extension.

Make sure that public-key authentication is allowed in the Connection Broker configuration, in the default settings and in the relevant connection profile (it is allowed by default). See [Section A.1.2.1.1](#) and [Section A.1.3.2](#).

To use the key pair for public-key authentication, you have to upload the public key to the remote host computer. If the remote host has an SFTP server running, you can automatically upload a copy of your new public key to the server with the wizard. To upload the key automatically, see [Section 5.4.3.2](#). To upload the key manually, see [Section 5.4.2](#).

### 5.4.3.1 Public-Key Generation

New keys are generated in the SSH Tectia Configuration tool. Select the **Keys and Certificates** page under **User authentication** and click **New Key** to start the Public-Key Authentication Wizard.

The image shows a screenshot of the 'Public-Key Authentication Wizard' dialog box. The window has a title bar with the text 'Public-Key Authentication Wizard' and standard window controls. The main area is divided into three sections. The first section, 'Enter properties for a new key', contains a 'File name' text box with the value 'key\_Administrator\_examplehost' and an empty 'Comment' text box. The second section, 'Enter a passphrase for the key', contains two text boxes: 'Passphrase' and 'Retype passphrase', both filled with dots. The third section, 'Key properties', is preceded by a 'Hide Options' button. It contains two dropdown menus: 'Key type' set to 'DSA' and 'Key length' set to '3072'. At the bottom of the dialog are three buttons: '< Back', 'Next >', and 'Cancel'.

**Figure 5.5. The Public-Key Authentication Wizard**

Define the key properties and the required passphrase to protect your key pair; you will be requested to enter the passphrase always when using the keys to authenticate yourself.

#### File Name

Type a unique name for the key file. SSH Tectia Client suggest a name consisting of the user name and the host name.

#### Comment

In this field you can write a short comment that describes the key pair. You can for example describe the connection the keys are used for. This field is not obligatory, but helps to identify the key later.

#### Passphrase

Type a phrase that you have to enter when handling the key. This passphrase works in a similar way to a password and gives some protection for your private key.



Make the passphrase difficult to guess. Use at least 8 characters, both letters and numbers. Any punctuation characters can be used as well.

Memorize the passphrase carefully, and do not write it down.

#### Retype passphrase

Type the passphrase again. This ensures that you have not made a typing error.

Click the **Advanced Options**, to define the type of the key to be generated and the key length to be different from the defaults. By default, SSH Tectia Client generates a pair of 2048-bit DSA keys.

In the **Key Properties** fields, you can make the following selections:

#### Key Type

Select the type of the key to be generated. Available options are DSA or RSA.

#### Key Length

Select the length (complexity) of the key to be generated. Available options are 1024, 2048 or 3072 bits. Larger keys are more secure, but also slower to generate.

Click **Next** to proceed to uploading the key as instructed in [Section 5.4.3.2](#).

### 5.4.3.2 Uploading Public Keys Automatically

Public keys can be uploaded automatically to servers that have the SFTP subsystem enabled. The **Public-Key Authentication Wizard** automatically uploads each new public key to a remote host of your choice. The wizard lists all existing keys, and you can select a key to upload it also to other remote servers at any time.

To access the **Public-Key Authentication Wizard**, click **User Authentication** → **Keys and Certificates** on the tree view.

Select a key and click **Upload**.

In the **Upload Public Key** view of the wizard, define the remote host where to upload the key:



**Figure 5.6. Uploading a key**

#### Quick connect

Select this option to define the remote **Host name** and your **user name** there. The default Secure Shell port is 22.

#### Connection profile

Select a **Connection profile** from the drop-down list that specifies the desired remote host and user name.

Click **Upload** to upload the key to the selected server. If you are already connected to the remote server host, the key upload starts immediately. If you are not connected, you will be prompted to authenticate on the server (by default with password).

The public key will be uploaded to the default user home directory (%USERPROFILE%\ .ssh2 on Windows, \$HOME/ .ssh2 on Unix).

### **Note**

The key user is required to have the `write` permissions to the key directory on the server, otherwise the automatic upload will fail. The administrator of the remote host computer may have restricted user access so that users are not able to configure public-key authentication for themselves even if public-key authentication is allowed in the server configuration.

Even if the automatic upload succeeds, it is possible that the server administrator has configured the system to store keys elsewhere than under the user home directory. In this case the keys and the authorization file additions have to be moved manually to the proper directory.

If you do not use the automatic upload facility, see [Section 5.4.2](#).

## 5.4.4 Using Keys Generated with OpenSSH

SSH Tectia Client supports also user key pairs generated with OpenSSH. The OpenSSH keys can be specified in the `ssh-broker-config.xml` file by using the `key-stores` element. An example configuration is shown below:

```
<key-stores>
  <key-store type="software"
    init="key_files(/home/exa/keys/id_dsa.pub,/home/exa/keys/id_dsa)" />
  <key-store type="software"
    init="directory(path(/home/exa/.ssh))" />
</key-stores>
```

This example adds a key called `id_dsa` and all keys from the user's default OpenSSH key directory (`.ssh` under the user's home directory).

You can add OpenSSH keys and directories on the **Keys and Certificates** page of the SSH Tectia Configuration tool. See [Section A.1.4.1](#).

The public key can be uploaded to the server the same way as with standard SSH2 keys. See [Section 5.4.2](#) and [Section 5.4.3.2](#).

## 5.4.5 Special Considerations with Windows Servers

If you use public-key authentication to log on to a Windows domain user account on SSH Tectia Server 5.1 or older, you must give your username as `DOMAIN\user` when attempting logon. On Unix command line, the backslash has to be escaped, for example:

```
$ sshg3 DOMAIN\\user@win-server
```

With SSH Tectia Server 5.2 and later, this is not required. When logging on to a machine that runs SSH Tectia Server 5.2 or later, and belongs to a Windows domain, the user account is by default assumed to be a domain account. If you want to log on to a local account instead, you have to specify the machine name as the "domain", for example on Windows command line:

```
> sshg3 MACHINE\user@machine
```

## 5.5 User Authentication with Certificates

Certificate authentication is technically a part of the public-key authentication method. The signature created with the private key and the verification of the signature using the public key (contained in the X.509 certificate when doing certificate authentication) are done identically with conventional public keys and certificates. The major difference is in determining whether a specific user is allowed to log in with a specific public key or certificate. With conventional public keys, every server must have every user's public key, whereas with

certificates the users' public keys do not have to be distributed to the servers - distributing the public key of the CA (self-signed certificate) is enough.

In brief, certificate authentication works in the following way:

1. The client sends the user certificate (which includes the user's public key) to the server. The packet also contains data unique to the session and it is signed by the user's private key.
2. The server uses the CA certificate (and external resources as required) to check that the user's certificate is valid.
3. The server verifies that the user has a valid private key by checking the signature in the initial packet.
4. The server matches the user certificate against the rules in the server configuration file to decide whether login is allowed or not.

### 5.5.1 Using the Configuration File (Unix)

To configure the client to authenticate itself with an X.509 certificate, perform the following tasks:

1. Enroll a certificate for yourself. This can be done, for example, with the **ssh-cmpclient-g3** or **ssh-scepclient-g3** command-line tools.

Example: Key generation and enrollment using **ssh-cmpclient-g3**

```
$ ssh-cmpclient-g3 INITIALIZE
-P generate://ssh2:passphrase@rsa:1536/user_rsa \
-o /home/user/.ssh2/user_rsa -p 62154:ssh \
-s 'C=FI,O=SSH,CN=user;email=user@example.org' \
-S http://fw.example.com:1080 http://pki.example.com:8080/pkix/ \
'C=FI, O=SSH, CN=Test CA 1'
```

For more information on **ssh-cmpclient-g3** and **ssh-scepclient-g3**, see [ssh-cmpclient-g3\(1\)](#) and [ssh-scepclient-g3\(1\)](#).

2. Place your keys and certificates in a directory where the Connection Broker can locate them.

By default, the Connection Broker attempts to use each key found in the `$HOME/.ssh2` directory on Unix, or in the `%APPDATA%\SSH\UserKeys` and `%APPDATA%\SSH\UserCertificates` directories on Windows.

You can also add other directory locations for keys on the **Keys and Certificates** page of the SSH Tectia Configuration tool. See [Section A.1.4.1](#). On Unix, you can use the `general/key-stores/key-store` element in the `ssh-broker-config.xml` file. See [the section called “Key Store Configuration Examples”](#).

3. (Optional) Create an identification file.

Using the `identification` file is not necessary if all your keys are stored in the default directory and you allow all of them to be used for public-key and/or certificate authentication. If the `identification`

file does not exist, the Connection Broker attempts to use each key found in the default directory. If the identification file exists, the keys listed in it are attempted first.

Specify the private key of your software certificate in the `$HOME/.ssh2/identification` file (the `CertKey` option works identically with the `IdKey` option):

```
CertKey      user_rsa
```

The certificate itself will be read from `user_rsa.crt`.

For more information on the syntax of the identification file, see [\\$HOME/.ssh2/identification](#).

4. Make sure that public-key authentication is enabled in the `ssh-broker-config.xml` file (it is enabled by default).

```
<authentication-methods>
  <auth-publickey />
  ...
</authentication-methods>
```

Other authentication methods can be listed in the configuration file as well. Place the least interactive method first.

## 5.5.2 Using the GUI

You can import existing PKCS #12, PKCS #7 and X.509 certificates on the **Keys and Certificates** page under User Authentication in the SSH Tectia Configuration tool. See [Section A.1.4.1](#).

## 5.6 Host-Based User Authentication (Unix)

Host-based authentication uses the public host key of the client machine to authenticate a user to the remote server. Host-based authentication can be used with SSH Tectia Client on Unix. The remote Secure Shell server can be either a Unix, Windows, or z/OS server.

Setting up host-based authentication usually requires administrator (root) privileges on the server. The setup is explained in the SSH Tectia Server documentation.

## 5.7 User Authentication with Keyboard-Interactive

Keyboard-interactive is a generic authentication method that can be used to implement different types of authentication mechanisms. Any currently supported authentication method that requires only the user's input can be performed with keyboard-interactive.

The supported submethods of keyboard-interactive depend on the Secure Shell server. Commonly supported submethods include password, RSA SecurID, RADIUS, and PAM authentication.



## Note

The client cannot request any specific keyboard-interactive submethod if the server allows several optional submethods. The order in which the submethods are offered depends on the server configuration. However, if the server allows, for example, the two optional submethods SecurID and password, the user can skip SecurID by pressing enter when SecurID is offered by the server. The user will then be prompted for a password.

### 5.7.1 Using the Configuration File (Unix)

To enable keyboard-interactive authentication on SSH Tectia Client, make sure that you have the following line in the `ssh-broker-config.xml` file:

```
<authentication-methods>
...
  <auth-keyboard-interactive />
...
</authentication-methods>
```

### 5.7.2 Using the GUI

Using keyboard-interactive authentication is a Connection Broker setting. Using the SSH Tectia Configuration tool to manage authentication methods is described in [Section A.1.3.2](#).

## 5.8 User Authentication with GSSAPI

GSSAPI (Generic Security Service Application Programming Interface) is a function interface that provides security services for applications in a mechanism independent way. This allows different security mechanisms to be used via one standardized API. GSSAPI is often linked with Kerberos, which is the most common mechanism of GSSAPI.

Kerberos libraries are installed by default on Linux platforms. They are also available for most other Unix platforms, but have to be installed separately.

For Windows, GSSAPI offers integrated authentication for Windows 2000/2003 networks with Kerberos. This method utilizes domain accounts, since local accounts are not transferable across machine boundaries.

The GSSAPI authentication method has no user interface (besides configuration). It does not ask anything from the user. If something fails during GSSAPI exchange, the reason for the failure can be seen in the client debug log.

## 5.8.1 Using the Configuration File (Unix)

To enable GSSAPI authentication on the client, make sure that you have the following line in the `ssh-broker-config.xml` file:

```
<authentication-methods>
  <authentication-method name="gssapi-with-mic" />
  ...
</authentication-methods>
```

Other authentication methods can be listed in the configuration file as well. Place the least interactive method first.

## 5.8.2 Using the GUI

Using the SSH Tectia Configuration tool to manage authentication methods is described in [Section A.1.3.2](#).





## Chapter 6 Transferring Files

SSH Tectia Client and SSH Tectia Server provide the basic secure file transfer functionality using the Secure File Transfer Protocol (SFTP).

For more advanced enterprise-class secure file transfer needs, we offer SSH Tectia ConnectSecure which is a separate product. It provides FTP-SFTP conversion, transparent TCP tunneling, enhanced file transfer (EFT) functionalities, and SFTP application programming interfaces (API), in addition to the basic Secure Shell client services. For more information on SSH Tectia ConnectSecure, see *SSH Tectia Client/Server Product Description* and *SSH Tectia ConnectSecure Administrator Manual*.

This chapter gives instructions on secure file transfer using the SCP and SFTP command-line tools and the SFTP graphical user interface (GUI).

### 6.1 Secure File Transfer with **scp3** and **sftpg3** Commands

SSH Tectia Client provides commands **scp3** (secure copy) and **sftpg3** for secure file transfer. These command-line clients apply the Secure File Transfer Protocol (SFTP).

When files are being uploaded with commands **scp3** and **sftpg3**, the files have the `TRUNCATE` flag on. The file size is shown as 0 until the file transfer has been completed.

These secure file transfer commands rely on the Connection Broker to take care of the cryptographic operations and authentication tasks, so they start the Connection Broker (the **ssh-broker-g3** process) in run-on-demand mode, if the Broker is not running already.

In case the **scp3** and **sftpg3** command-line clients are used in scripts that start several file transfer commands at the same time, the Connection Broker must already be running in the background. Since the Connection Broker takes a few seconds to become up and running, make sure the scripts are not started immediately, because they can fail if the Connection Broker is still starting.

To start the Connection Broker, run the **ssh-broker-g3** command. For more information, see [ssh-broker-g3\(1\)](#).

### 6.1.1 Using scp3

**scp3** (**scp3.exe** on Windows) is used to securely copy files over the network. **scp3** uses **ssh-broker-g3** to provide a secure transport using the Secure Shell version 2 protocol. The remote host(s) must be running a Secure Shell version 2 server with the **sftp-server** (or **sft-server-g3**) subsystem enabled.

The basic syntax of **scp3** is:

```
scp3 user@source:/directory/file user@destination:/directory/file
```

**scp3** can be used to copy files in either direction; from the local system to the remote system or vice versa. Copies between two remote hosts are also permitted. Local paths can be specified without the *user@system:* prefix. Relative paths can also be used, they are interpreted in relation to the user's home directory.

Windows paths should be preceded by a slash ("/"). For example, copying a local file to a remote Windows server:

```
scp3 localfile user@destination:/C:/directory/file
```

For more information on the command-line options, see [scp3\(1\)](#).

### 6.1.2 Using sftpg3

**sftpg3** (**sftpg3.exe** on Windows) is an FTP-like client that can be used for secure file transfers over the network. **sftpg3** uses **ssh-broker-g3** to provide a secure transport using the Secure Shell version 2 protocol.

Even though it functions like **ftp**, **sftpg3** does not use any FTP daemon or FTP client for its connections. **sftpg3** can be used to connect to any host that is running a Secure Shell version 2 server with the **sftp-server** (or **sft-server-g3**) subsystem enabled.

The basic syntax of **sftpg3** is:

```
sftpg3 user@host
```

**sftpg3** has two connection end points, local and remote, and both of them can be connected to other hosts than the SSH Tectia Client host. By default, the local end point is connected to the filesystem of the SSH Tectia Client host and the remote end point is connected to the host defined on the command line (or left unconnected if no host is defined on the command line).

When started interactively, **sftpg3** displays a prompt where the SFTP commands can be entered, much like in the traditional **ftp** program. It is also possible to start **sftpg3** non-interactively with a batch file that contains the commands to be run.

For more information on the command-line options and commands, see [sftpg3\(1\)](#).

## 6.2 Secure File Transfer GUI (Windows)

SSH Tectia Client on Windows provides a secure file transfer GUI that makes it easy to download files from a remote host computer into your local computer and to upload files to a remote host. The SSH Tectia file transfer window operates much like Windows Explorer.

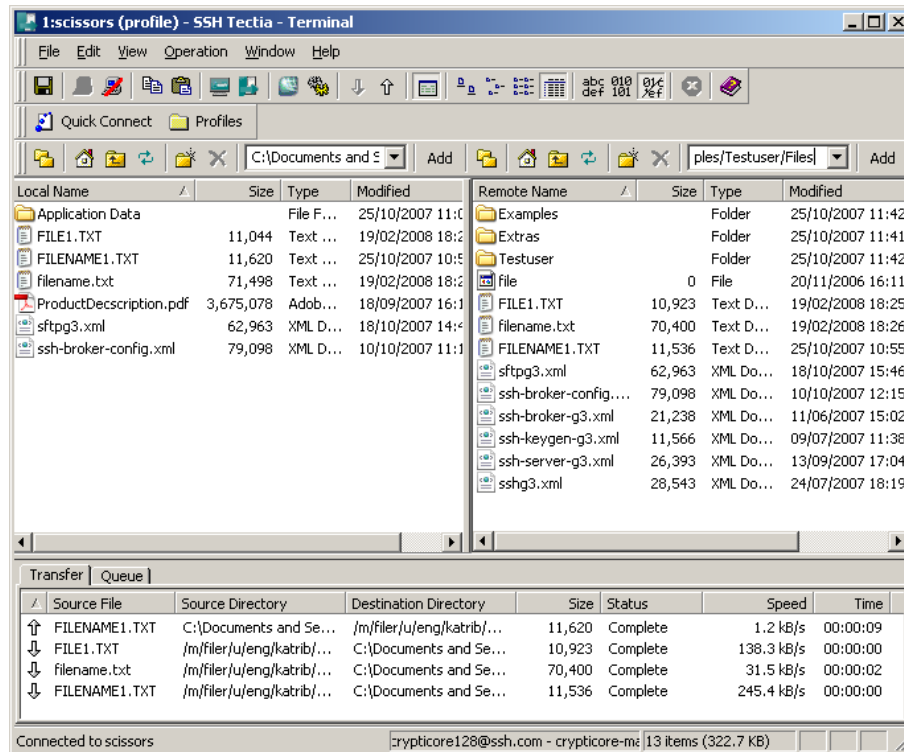


Figure 6.1. SSH Tectia File Transfer GUI

In the SSH Tectia file transfer window, you can use the connection profiles defined in the Connection Broker configuration, or connect to a remote host using the **Quick Connect** option.

### 6.2.1 Defining File Transfer GUI Settings

Configuring the default settings for the SSH Tectia File Transfer GUI is explained in [Section B.1.5](#).

### 6.2.2 Downloading Files with the File Transfer GUI

There are different ways to download a file, or several files at the same time. Selecting multiple files with the Shift or Control key works the same way as in Windows Explorer.

### Drag and drop

Dragging and dropping is probably the easiest way to download files. Simply select the file(s) you want to download, hold down the mouse button and move the file to a location where you want it - for example on the Windows desktop - and release the button.

### Download button

Click the **Download** button on the toolbar to download the selected file(s).

### Shortcut menu

When you right-click a file in the Remote View, a shortcut menu appears. Select **Download** to perform the transfer immediately, or select **Download Dialog** to define another destination folder for the files.

If you selected the **Download Dialog** option, a **Download - Select Folder** dialog appears. This is a standard Windows file selection dialog, where you can select the location where you want the selected file(s) to be downloaded. After you have selected the appropriate folder (or some other location), the current downloading status will be shown in the Transfer View.

## 6.2.3 Uploading Files with the File Transfer GUI

The file transfer window can be used to upload files from your local computer to the remote host computer. There are different ways to upload a file, or several files at the same time. Selecting multiple files with the Shift or Control key works the same way as in Windows Explorer.

### Drag and drop

Dragging and dropping is probably the easiest way to upload files. Simply click on the local file(s) you want to upload (for example on the desktop or Windows Explorer), hold down the mouse button, move the file(s) into the file view in the **File Transfer** window, and release the button.

### Upload button

Click the **Upload** button on the file transfer window toolbar to upload the selected file(s).

### Shortcut menu

When you right-click a file in the Local View, or an empty space in the Remote View, a shortcut menu appears. Select the **Upload** to perform the transfer immediately, or select **Upload Dialog** to select the files to upload.

If you have selected the **Upload Dialog** option, an **Upload - Select Files** dialog appears. This is a standard Windows file selection dialog, where you can select which file(s) you want to upload. After you have selected the files, the Transfer View shows the current uploading status.

## 6.2.4 Transfer and Queue Tabs

At the bottom of the SSH Tectia File Transfer view, there are two tabs: **Transfer** and **Queue**. See [Figure 6.1](#).

The **Transfer** tab displays a list of files that have already been transferred between the local and remote computers.

The **Queue** tab can be used to create a customized list of files that will be uploaded or downloaded at a later stage. You can use the mouse to drag and drop files to the Queue tab, where they will wait for a separate transfer command.

After logging in to a remote host, right-clicking the **Queue** page opens a shortcut menu with the following options:

- **Add** for adding selected files to the queue, right-click on the Queue page and select **.**

The **Edit Transfer Queue** dialog appears. Click **New** above the list area to type in the path to a new file to be transferred, or click the ellipsis button ( . . . ) to browse to the files.

- To edit the target locations of the queued files, select a file to edit, right-click the Queue page and select **Edit**.

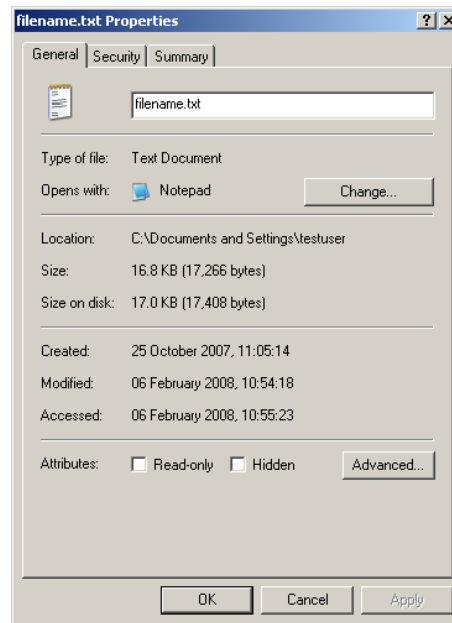
The **Edit Transfer Queue** dialog opens, allowing you to type in a new destination directory for the file. You can also click the ellipsis button ( . . . ) to browse to the destination directory.

You can use the **Edit** option for several files at the same time, but the direction of the transfer (upload or download) must be the same for all of the files.

- To delete files from the queue, select the files, right-click the Queue page and select **Remove**.
- To transfer single files, select them, right-click the Queue page and select **Transfer**.
- To transfer all the queued files, right-click the Queue page and select **Transfer All**.

## 6.2.5 Defining File Properties

Selecting a file in the Local View or Remote View and selecting **Operation** → **Properties** (or **Properties** on the shortcut menu) opens the File Properties dialog which allows you to view and change some of the file properties.



**Figure 6.2. Properties page for a file**

View and define the properties as necessary. You can modify for example the read-write permissions of the file, or define the file as hidden.

For more information, see the operating system documentation.

## 6.2.6 Differences from Windows Explorer

The file transfer window operates very much the same way as Windows Explorer. However, due to the different nature of handling files locally in your own computer (as per Windows Explorer) and handling them over a secured remote connection in the host computer (as per SSH Tectia Client file transfer), there are some differences in operation.

### Deleting folders

It is not possible to delete a remote folder that is not empty. Delete the files and subfolders in the folder first.

### Multiple paste operations

During copy and paste operations, the file names cannot be changed when the files are pasted. Therefore it is not possible to paste files several times into one location, creating multiple copies of the pasted files as in Windows Explorer.



### Note

The maximum size of transferred files is limited only by the file system. (On many systems the maximum file size is 2 gigabytes.)

## 6.3 Controlling File Transfer

The current Secure File Transfer Protocol (SFTP) does not transfer any information about the files to be transferred, only the file contents as a byte stream. This is sufficient for Unix-type files if the sender and receiver use the same CCS.

With MVS datasets, SSH Tectia needs more information: which transfer format to use, what code sets are involved, and what the file characteristics are. SSH Tectia introduces some extensions to SFTP and the information can be relayed by using the Site commands of **scp3** and **sftp3**.

For more information on MVS file transfers, see *SSH Tectia Server for IBM z/OS User Manual*.

### 6.3.1 Site Command

The Site commands are the recommended way for controlling file transfer when both the client and server host are running SSH Tectia.

For command descriptions, see the **site** and **lsite** command in [sftp3\(1\)](#) and the `--dst-site` and `--src-site` options in [scp3\(1\)](#).

When giving the command, either the abbreviation or the full command can be used. For example, the following two commands accomplish the same thing:

```
sftp> site X=bin
sftp> site transfer_mode=bin
```

The available **SITE** parameters are:

A | transfer\_translate\_dsn\_templates=TEMPLATES

*TEMPLATES* specifies the search templates for the translate table. Write '%T' to show the point where the translate table name (see above) is to be inserted. Delimit the templates with a plus character. The dataset name templates must not contain slashes, instead they must be preceded by two or three underscores.

The first translate table dataset that is found is used to perform the code conversion.



#### Note

The translate table must translate line delimiters into EBCDIC NL characters. See [F | transfer\\_format=FORMAT](#).

Default: none

automount=YES | NO | IMMED

If set to YES and a normal allocation fails because a dataset is not online, SSH Tectia will allocate it and request the system to mount it. This requires that the user has read permission to the `SSZ.MOUNT` facility.

If set to NO, offline datasets are not mounted automatically.

If set to `IMMED`, SSH Tectia will not attempt the normal allocation, it will request the system to mount the dataset immediately.

Default: `no`

`AUTOMount`

Equal to `automount=yes`.

`autorecall=YES|NO`

Defines whether datasets migrated by a storage manager are recalled automatically.

Default: `yes`

`AUTORECALL`

Equal to `autorecall=yes`.

`B|BLKsize|BLOCKSize=SIZE`

Maximum block size.

Default: `none`

`BLocks`

Specifies that the space allocation unit is blocks. Equal to `space_unit=blks`.

`C|transfer_codeset=CODESET`

During the transfer the data has the specified codeset. `CODESET` is the codeset name that is known to the **iconv** function of the system performing the conversion. The available codesets can be listed by invoking the **iconv** command at a USS prompt with the `-l` option:

```
$ iconv -l
```

Default: `none`

In the following example, a Windows SFTP client puts a file to a z/OS dataset and gets a dataset from z/OS:

```
sftp> site C=ISO8859-1 D=IBM-1047
sftp> sput file.txt //DATASET.TXT
sftp> sget //DATASET.TXT file.txt
```

The **site** command tells the z/OS server that the codeset during transfer is ISO8859-1 and that the dataset is stored on the server with the IBM-1047 codeset. In **sput**, this means that the server converts the codeset from ISO8859-1 to IBM-1047 upon receiving the data. In **sget**, this means that the server converts the codeset from IBM-1047 to ISO8859-1 before sending the data.



## Note

The codeset information is always given to the host that is capable of performing the conversion, in this case the z/OS host.



`CONDdisp=CATLG|UNCATLG|KEEP|DELETE`

Specify the disposition of the output file when a file transfer ends prematurely.

If set to `CATLG`, the output file is kept when a file transfer ends prematurely. If the output file is a MVS dataset, it is cataloged.

If set to `KEEP`, the output file is kept. If it is a MVS dataset it is not cataloged.

If set to `DELETE`, the output file is deleted when a file transfer ends prematurely. If it is a MVS dataset, it is also uncataloged.

If set to `UNCATLG`, the output file, if it is a HFS file, is deleted. If it is a MVS dataset, the dataset is kept and uncataloged.

Default: `catlg`

`Cylinders`

Specifies that the space allocation unit is cylinders. Equal to `space_unit=cyls`.

`D|transfer_file_codeset=CODESET`

The data in the dataset has the specified codeset. `CODESET` is the codeset name that is known to the **iconv** function of the system performing the conversion. The available codesets can be listed by invoking the **iconv** command at a USS prompt with the `-l` option:

```
$ iconv -l
```

Default: none

`DATAclass|dataclas=CLASS`

Specifies the data class of a dataset.

Default: none

`dataset_sequence_number=NUMBER`

Identifies the relative position of a dataset on a tape volume.

Default: System default

`defer=YES|NO`

Specify whether dataset allocation is postponed from allocation phase to opening the dataset.

If set to `yes` dataset allocation is postponed until dataset is opened.

If set to `no` dataset is allocated in allocation phase.

Default: `no`

`defer`

Specifies that dataset allocation is postponed until dataset is opened. Equal to `defer=yes`.

E|transfer\_translate\_table=TABLE

TABLE is the name of the table that specifies the codeset conversion. If set, this attribute overrides the transfer codeset and file codeset attributes. The table is always applied in the normal direction, that is, the first character array is used for incoming (from the line to the dataset) data and the second array for outgoing data. If the opposite translation is needed, e.g. the dataset contains ASCII and should be transferred as EBCDIC, the users (or their system programmer) can prepare a table dataset with the character arrays in reversed order (e.g. with the system utility CONVXLAT or by editing an existing translate dataset).

expiry\_date=YYDD/YYYDD

Specifies the expiration date for a new dataset. On and after this date, the operating system can delete or write over the dataset.

Default: System default

F|transfer\_format=FORMAT

The byte stream consists of the bytes that are transferred as payload in the SFTP protocol packets. The byte stream has one of the following formats: **stream**, **line**, or **record**. All three formats may have data consisting of text, non-text data, or a mixture of these.

When writing an MVS dataset, a record that is longer than the maximum or fixed record length will cause an error unless `record_truncate` is set to `yes`, in which case it will be truncated. When writing to datasets with fixed record lengths, short records will be filled with binary zeroes if you use the record transfer format and with blanks if you use the line transfer format.

- **F=stream**: The stream transfer format contains the data bytes of the dataset but no structural information. If a dataset with a fixed record length is transferred with the stream format and recreated with the same record length, the record structure will be preserved. Variable length records will not be recreated properly if transferred with the stream format.
- **F=line**: The line transfer format is record-based. It uses delimiter characters to mark the end of a record. The delimiter character may be a Carriage Return or a Newline. When writing to or reading from datasets with ASA control characters, a Form Feed is also treated as a delimiter. The table below shows the values of these characters in EBCDIC and ASCII. Data sent to SSH Tectia Client in the line transfer format must be in EBCDIC or must be converted to EBCDIC during the transfer.

Delimiter	EBCDIC				ASCII			
	Name	Dec	Oct	Hex	Name	Dec	Oct	Hex
\r Carriage Return	CR	13	015	0x0D	CR	13	015	0x0D
\n Newline	NL	21	025	0x15	LF	10	012	0x0A
\f Form Feed	FF	12	014	0x0C	FF	12	014	0x0C

Note that ASCII does not have a NL character, instead LF is used to delimit lines.

Avoid conversions that transform an ASCII Line Feed (LF/10/012/0x0A) into an EBCDIC Line Feed (LF/37/045/0x25) or an EBCDIC Newline (NL/21/025/0x15) into an ASCII Next Line (NEL/133/0205/0x85).

Be aware that sending a double delimiter, e.g. `\r\n` or `\n\r`, to SSH Tectia Client will result in two records. The `transfer-line-delimiter` and `file-line-delimiter` advice string attributes can be used to cause the SSH Tectia Client server or client program to convert between the line delimiter conventions.

SSH Tectia Client sends `\n` as the Server Newline Convention in the server initialization SFTP protocol message.

When transferring line format data to and from MVS files with ASA line printer control characters, SSH Tectia Client will convert between the control characters and line delimiter characters, as described in the IBM document *z/OS C/C++ Programming Guide, SC09-4765-03*, Chapter 8.

To transfer records without changing the ASA code, use the `stream` or `record` transfer format, or define the dataset using a DD card and specify `RECFM=FB` or `RECFM=VB`.

Datasets transferred in the line transfer format and recreated on a mainframe will not necessarily be identical.

- `F=record`: The record transfer format is record-based. Each record is preceded by a length field consisting of a 4-byte big-endian binary integer, which indicates the number of data bytes in the record. Note that the format is not the same as the record descriptor word in datasets with `RECFM=V` or `VB`.

A dataset that is transferred with the record transfer format can be recreated as any dataset type.

Default: `line`.

`file_status=NEW|MOD|SHR|OLD`

Defines the status of a dataset. If entered, the value will be used when allocating the dataset. This attribute corresponds to the first value in the `DISP` parameter of the JCL DD statement.

Default: `NEW` for datasets that will be created, `SHR` for datasets that will be read only, otherwise `OLD`.

`fixrecfm=LENGTH`

The dataset organization is set to `FB` and the fixed record length is set to `LENGTH`.

Default: `none`

`I|transfer_line_delimiter=CONVENTION`

The transfer line delimiter specifies the newline convention used during the file transfer. Possible values are:

- `I=mvs`: The line delimiter during the transfer is NL (`\n`, `0x0a`).
- `I=unix`: The line delimiter during the transfer is NL (`\n`, `0x0a`).
- `I=dos`: The line delimiter during the transfer is LFNL (`\r\n`, `0x0d0a`).
- `I=mac`: The line delimiter during the transfer is LF (`\r`, `0x0d`).

Default: none

In the following example, a Windows SFTP client puts a file to a z/OS dataset and gets a dataset from z/OS:

```
sftp> site I=dos J=mvs
sftp> sput file.txt //DATASET.TXT
sftp> sget //DATASET.TXT file.txt
```

The **site** command tells the z/OS server that the line delimiter during the transfer is LFNL and that the dataset is stored with the NL line delimiter. In **sput**, this means that the server converts the line delimiters from LFNL to NL upon receiving the data. In **sget**, this means that the server converts the line delimiters from NL to LFNL before sending the data.

### Note

The line delimiter information is always given to the host that is capable of performing the conversion, in this case the z/OS host.

**J**|transfer\_file\_line\_delimiter=CONVENTION

The transfer file line delimiter specifies the newline convention used in the (source or destination) file.

Possible values are:

- **J=mvs**: The line delimiter used in the file is NL (\n, 0x0a).
- **J=unix**: The line delimiter used in the file is NL (\n, 0x0a).
- **J=dos**: The line delimiter used in the file is LFNL (\r\n, 0x0d0a).
- **J=mac**: The line delimiter used in the file is LF (\r, 0x0d).

Default: none

**keylen**=LENGTH

Specifies the length in bytes of the keys used in the dataset.

Default: none

**keyoff**=OFFSET

Specifies the key offset. The position of the first byte of the key in records of the specified VSAM dataset.

Default: none

**L**|size=SIZE

Size estimate in bytes for dataset allocation.

Default: 1000000

`label_type=SL|NSL|SUL|LTM|AL|AUL`

The type of the label for the dataset. This attribute corresponds to the first value in the `LABEL` parameter of the JCL DD statement.



## Note

The values `NL` (no label) and `BLP` (bypass label processing), which can be specified in JCL, are not allowed for this attribute in SSH Tectia.

`like=LIKE`

Specifies the name of a model dataset from which the `RECFM`, `BLKSIZE`, and `LRECL` attributes are to be copied. The name must be the full DSN of a cataloged dataset and must be preceded with three underscores.

You must include the `type` attribute when using `like` unless you are creating a PS dataset and the model is a PS dataset.

Default: none

`M|Directory|directory_size=SIZE`

Number of 256-byte records in the directory.

Default: 10

`MGmtclass|mgmtclas=CLASS`

Specifies the management class of a dataset.

Default: none

`NOAUTOMount`

Equal to `automount=no`.

`NOAUTOREcall`

Equal to `autorecall=no`.

`NORMdisp=CATLG|UNCATLG|KEEP|DELETE`

Specifies the dataset disposition to be used after a file transfer that ends normally. This attribute corresponds to the second value in the `DISP` parameter of the JCL DD statement.

Default: `CATLG`

`NOTRAILINGblanks`

Equal to `trailing_blanks=no`.

`NOTRUNcate`

Equal to `record_truncate=no`.

O|*RECFM=RECFM*

*RECFM* specifies the dataset organization. The possible values are all valid combinations of the following letters:

F	Fixed
V	Variable
U	Undefined
B	Blocked
S	Spanned or standard
M	Machine line printer codes
A	ASA line printer codes

Default: *vb*

P|*profile=PROFILE*

The file transfer profile specifies the named profile used for the file transfer. The profile name is case-sensitive. With special profile name *P=%* no profiles are used. This also prevents profile matching based on file name.

Default: none

PRImary|*primary\_space=SPACE*

Primary space allocation for a dataset.

Default: none

R|*LRecl=LENGTH*

Maximum record length or fixed record length.

Default: 4096, for VSAM, 80, if dataset organization is F or FB, otherwise 1024

*retention\_period=DAYS*

The retention period in days. After the retention period, the dataset expires and the operating system can delete or write over the dataset.

Default: System default

S|*staging=NO|YES*

Specify whether staging is to be used in the SFTP server when accessing a file or dataset.

If set to *no*, staging is not used.

If set to *yes*, staging is used, when needed.

If this parameter is not set, the server decides whether staging will be used or not.

S|*staging=NO|YES*

Specify whether staging is to be used in the SFTP server when accessing a file or dataset.

If set to `no`, staging is not used.

If set to `yes`, staging is used, when needed.

If this parameter is not set, the server decides whether staging will be used or not.

`SECondary|secondary_space=SPACE`

Secondary space allocation for a dataset.

Default: none

`space_unit=UNIT`

Unit of space allocation for a dataset.

Possible values are:

- `space_unit=blks`: Allocation unit is blocks.
- `space_unit=cyls`: Allocation unit is cylinders.
- `space_unit=trks`: Allocation unit is tracks.
- `space_unit=avgreclen`: Allocation unit is average record length.

Default: none

`space_unit_length=LENGTH`

When `space_unit=blks` or `space_unit=avgreclen`, specifies the size of the space allocation unit.

Default: 100 with `space_unit=avgreclen`, none with `space_unit=blks`

`STOrclass|storclas=CLASS`

Specifies the storage class of system managed storage.

Default: none

`svc99_text_units=STRING`

Dynamic allocation arguments that override or are added to arguments from other file transfer attributes.

Default: none

`T|type=TYPE`

The file type specifies the type of the dataset when the dataset is created. The available values are:

- `T=hfs`: The type of the created dataset is HFS.
- `T=po`: The type of the created dataset is PDS.
- `T=poe`: The type of the created dataset is PDSE.

- `T=esds`: The type of the created dataset is VSAM ESDS.
- `T=ksds`: The type of the created dataset is VSAM KSDS.
- `T=rrn`: The type of the created dataset is VSAM RRN.
- `T=ps`: The type of the created dataset is PS.

Default: `po`, if dataset name includes member, otherwise `ps`

#### TRacks

Specifies that the space allocation unit is tracks. Equal to `space_unit=trks`.

#### trailing\_blanks=YES|NO

Specify whether to preserve trailing blanks in a transferred dataset.

If set to `yes` trailing blanks will be transferred. This can be used, for example, to preserve the structure of fixed format datasets.

If set to `no` trailing blanks will be stripped.

Default: `no`

#### TRAILingblanks

Equal to `trailing_blanks=yes`.

#### TRUNcate

Equal to `record_truncate=yes`.

#### U|record\_truncate=YES|NO

When a record truncation occurs while writing an MVS dataset, the system will continue writing the dataset if `record_truncate` is set to `yes`; and the system will abort the transfer if `record_truncate` is set to `no` or omitted.

Record truncation will occur if the length of a transferred record (after codeset and line delimiter conversion) is larger than the maximum record length of the dataset. Truncation can occur only when the transfer format is `line` or `record`. Note that the `stream` format does not have any concept of records in transferred data and it will fill out all records to their maximum length.

In the `line` transfer format, the length of a transferred record is the number of characters up to a newline character.

In the `record` format, the length of a transferred record is given by the 4 byte binary length field which precedes the record.

The maximum length of a dataset record depends on the dataset organization:

```
F and FB - LRECL
V and VB - LRECL-4
```



U	- BLKSIZE
VSAM	- MAXRECLEN

When SSH Tectia Client aborts writing a dataset because of record truncation, it will complete the write operation during which the system observed the truncation. It will write to disk one or more records, at least one of which is truncated. The dataset is left on the system.

SSH Tectia Client may write a large amount of data in one write operation, typically 32kB. Several records may be written in the last operation, some of them truncated. Small files may be written to the end of the file, and thus the resulting dataset will be equivalent to one written with setting `record_truncate=yes`.

Note that some file transfer client programs do not always show the error or warning messages from the server. Using the verbose mode (`--verbose, -v`) may show more messages from the server.

## Note

When SSH Tectia Client writes a dataset with `record_truncate=yes`, data loss may occur.

`unit=UNIT`

The name of device or group of devices that the dataset will reside on (or does reside on, if it already exists). The maximum length of `UNIT` is 8 characters. If the value exceeds the maximum length, it is truncated to 8 characters.

It is also possible to specify a device address. Precede a four digit address with an underscore.

Default: none

`unit_count=NUMBER`

Specifies the number of devices for the dataset. This attribute corresponds to the second value in the `UNIT` parameter of the JCL DD statement.

Default: System default

`unit_parallel=YES|NO`

Asks the system to mount all the volumes for the dataset in parallel. This attribute corresponds to the character 'p' in the second value in the `UNIT` parameter of the JCL DD statement.

Default: System default

`volumes=VOLUMES`

A plus sign (+) separated list of volumes a dataset will reside on (or does reside on, if it already exists).

Default: none

`volume_count=NUMBER`

Specifies the maximum number of volumes that an output dataset requires. This attribute corresponds to the volume count value in the `VOLUME` parameter of the JCL DD statement.

Default: System default

`x|transfer_mode=MODE`

The transfer mode specifies whether codeset and line delimiter conversions are performed. The available values are:

- `x=bin`: Codeset and line delimiter conversions are not performed.
- `x=text`: Codeset and line delimiter conversions are performed.

Default: none



## Note

If `transfer_mode` is not given but both `transfer_codeset` and `transfer_file_codeset` or `transfer_translate_table` are present conversions are performed.

## Chapter 7 Secure Shell Tunneling

Tunneling is a way to forward otherwise unsecured application traffic through Secure Shell. Tunneling can provide secure application connectivity, for example, to POP3, SMTP, and HTTP-based applications that would otherwise be unsecured.

The Secure Shell v2 connection protocol provides channels that can be used for a wide range of purposes. All of these channels are multiplexed into a single encrypted tunnel and can be used for tunneling (forwarding) arbitrary TCP/IP ports and X11 connections.

The client-server applications using the tunnel will carry out their own authentication procedures, if any, the same way they would without the encrypted tunnel.

The protocol/application might only be able to connect to a fixed port number (e.g. IMAP 143). Otherwise any available port can be chosen for tunneling. For remote tunnels, the ports under 1024 (the well-known service ports) are not allowed for ordinary users, but are available only for system administrators (root privileges).

There are two basic kinds of tunnels: local and remote. They are also called outgoing and incoming tunnels, respectively. X11 forwarding and agent forwarding are special cases of a remote tunnel. The different tunneling options are handled in the following sections.

### 7.1 Local Tunnels

A local (outgoing) tunnel forwards traffic coming to a local port to a specified remote port.

With `sshg3` on the command line, the syntax of the local tunneling command is as follows:

```
client$ sshg3 -L [protocol/] [listen-address:]listen-port:dst-host:dst-port sshserver
```

where:

- `[protocol/]` specifies which protocol is to be used in the tunneled connection, it can be `ftp` or `tcp` (optional argument). The default is `tcp`.

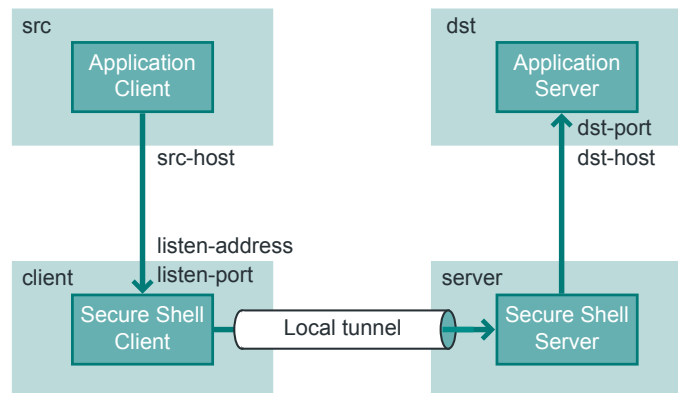
- `[listen-address:]` defines which interface on the remote server will be listened to (optional argument). By default all interfaces are listened.
- `listen-port` is the number of the port on the remote server, and connections coming to this port will be tunneled to the client.
- `dst-host:dst-port` define the destination host address and the port to which the connection is tunneled from the client.
- `sshserver` is the IP address or the host name of the Secure Shell server.

Setting up local tunneling allocates a listener port on the local client host. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the remote server and another connection is made from the server to a specified destination host and port. The connection from the server onwards will not be secure, it is a normal TCP connection.

## **i** Note

Every user with access to the local client host will be able to use the local tunnels.

Figure 7.1 shows the different hosts and ports involved in local tunneling (port forwarding).



**Figure 7.1. Local tunneling terminology**

For example, when you issue the following `sshg3` command on the command line, all traffic coming to port 1234 on the client host will be forwarded to port 23 on the server.

```
client$ sshg3 -L 1234:localhost:23 --abort-on-failing-tunnel username@sshserver
```

The forwarding address in the command is resolved at the (remote) end point of the tunnel. In this case `localhost` refers to the server host (`sshserver`).

In this example, also the `--abort-on-failing-tunnel` option is specified. It causes the command to abort if creating the tunnel listener fails (for example, if the port is already reserved). Normally if the connection to the server succeeds, but creating the listener fails, no error message is given.

## 7.1.1 Transparent TCP Tunneling

Transparent TCP Tunneling is an optional feature. You need to select it on separately at the installation phase. For detailed information on the supported versions, see *Product Specification* in *SSH Tectia Client/Server Solution Product Description*.

The transparent TCP tunneling feature captures the TCP traffic of hosts defined in the SSH Tectia configuration and uses encrypted tunnels for sending the data. No changes are required to the configuration of the application software.

The transparent TCP tunneling is activated in the Connection Broker configuration, where you can also specify the applications to be tunneled and define filter rules that control the setting up of the tunnels in detail. Once activated, the transparent TCP tunneling feature automatically captures the defined applications and the Connection Broker creates Secure Shell tunnels to the defined servers, that can be SSH Tectia Servers, SSH Tectia Server for IBM z/OS, or any SSH2-capable Secure Shell servers.

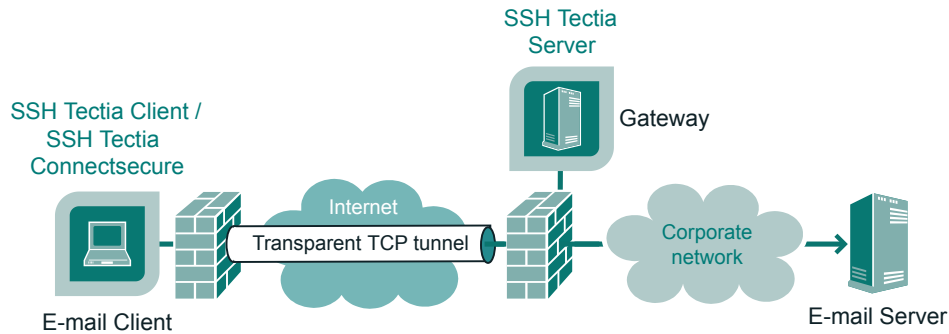
For information on the configuration settings in the XML file, see [the section called “The filter-engine Element”](#), and for settings in the GUI, see [Section A.1.6](#).

When a global configuration file exists, (for example when SSH Tectia Client is controlled by SSH Tectia Manager,) and it includes the `filter-engine` element, those settings are applied. The global configuration file is located in `/etc/ssh2/ssh-broker-config.xml` on Unix, and `"C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Broker\ssh-broker-config.xml"` on Windows.

### 7.1.1.1 Example on Tunneling E-mail Service (Windows)

This section gives an example of configuring encrypted tunnels for an e-mail service on Windows. Transparent TCP tunneling is used for establishing tunnels that can be utilized as a secure transport between an e-mail client and an e-mail server communicating over the Internet.

This scenario describes a typical configuration for remote users for accessing the company's internal e-mail services transparently. In the test scenario, access from the client's private network to the Internet traverses through a SOCKS4 server, and the client-side has SSH Tectia Client installed. Access to the company's internal network, including the e-mail services, goes via a gateway host which has SSH Tectia Server running.



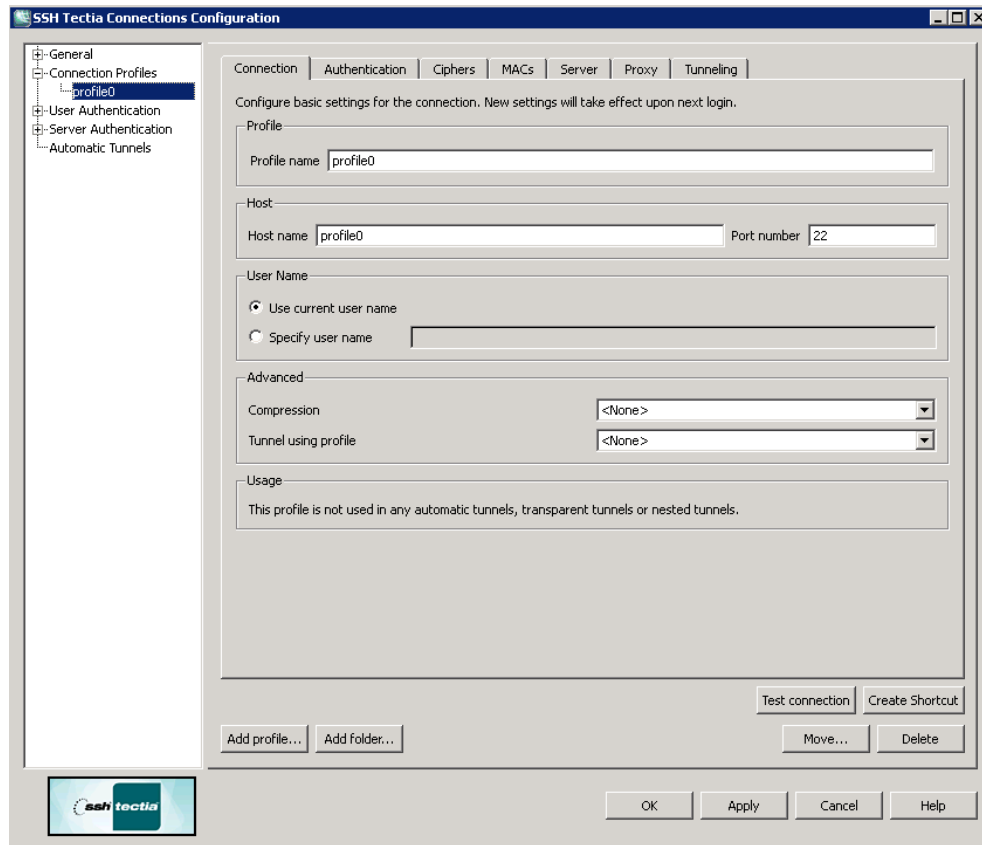
**Figure 7.2. Transparent TCP tunneling securing e-mail connections**

Before e-mail delivery, SSH Tectia Client automatically creates a transparent TCP tunnel between the client host and the SSH Tectia Server gateway for SMTP/IMAP/POP protocols. The encrypted tunnel ends at the gateway, and from there onwards the e-mail traffic is transmitted unencrypted in the company's internal network.

To create the configuration using the SSH Tectia Configuration tool, do the following:

1. On the **Connection Profiles** page, click **Add profile** to add a new connection profile for the gateway server host. Enter the profile name and click **OK**. In the example, the profile is named `paper`.

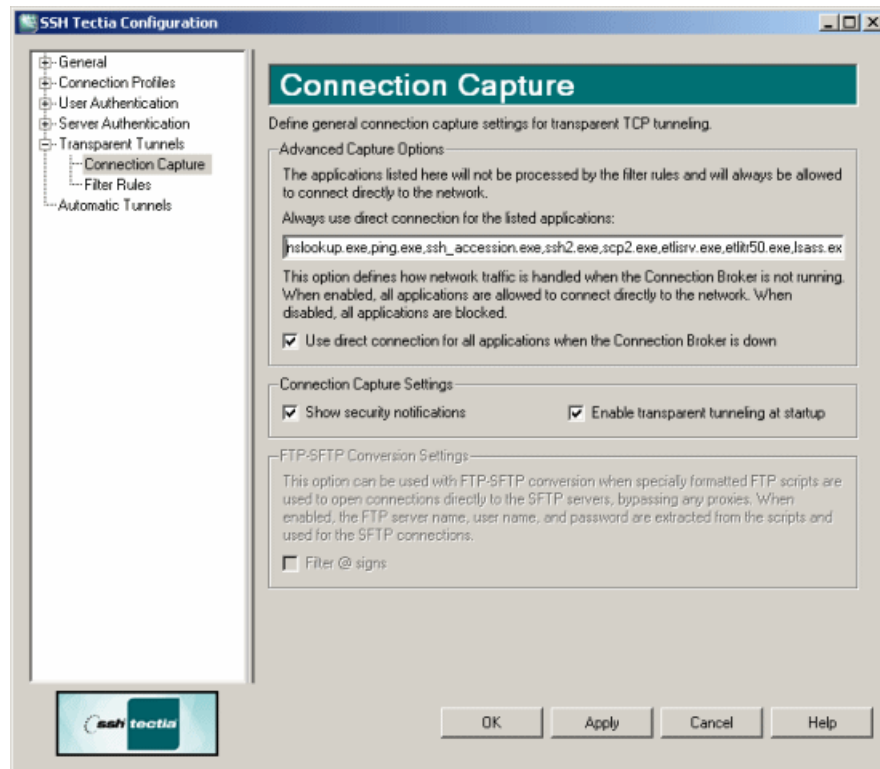
Select the created profile from the **Connection Profiles** list, and specify the connection details in the following view.



**Figure 7.3. Configuring connection profiles**

By default, the profile name is assumed to be the host name of the destination server, but here we link the profile name to a server called `host1234`. Click **Apply** when the settings are ready.

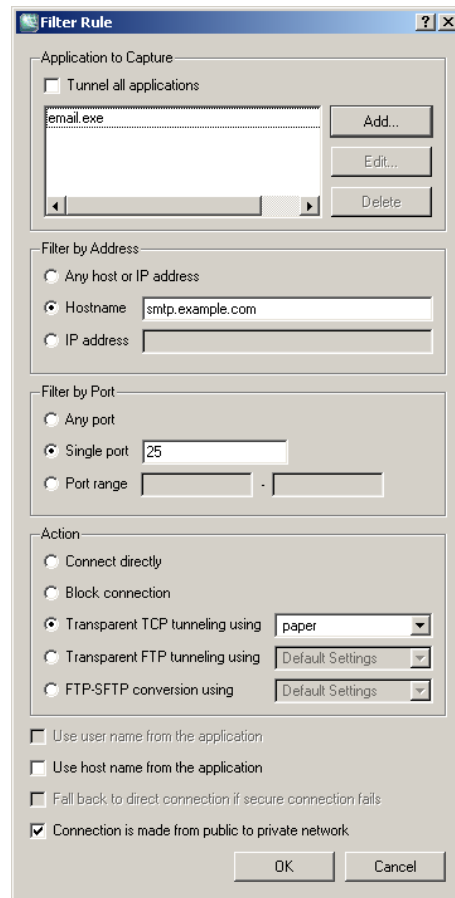
2. On the **Transparent Tunnels** → **Connection Capture** page, select the **Enable transparent tunneling at startup** check box.



**Figure 7.4. Defining the transparent TCP tunneling settings**

3. On the **Transparent Tunnels** → **Filter Rules** page, click **Add** to add the filter rules which define the applications and ports to be captured and tunneled. In this example, only TCP ports related to e-mail delivery (IMAP, POP, SMTP) are forwarded to the gateway server through profile `paper`.

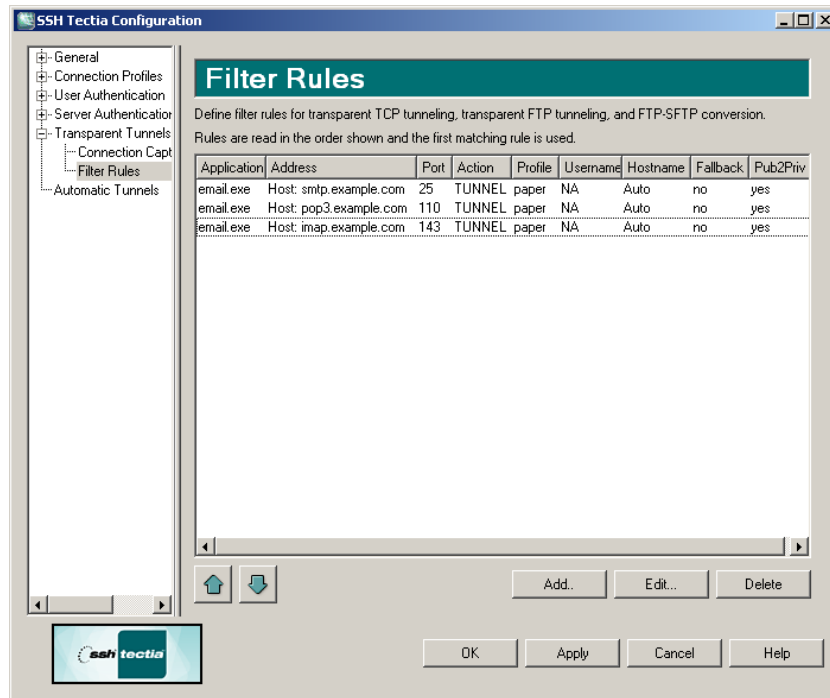




**Figure 7.5. Adding filter rules for email services**

Under **Action**, it is possible to select how the destination host is defined; either the host defined in the connection profile is used, or the host definition is received from the application. Now that we have a gateway server in this example, we will use the hostname defined in the profile, and the **Use host name from the application** is left unselected. In case we had a Secure Shell server running on each destination server, we could use the host names from the application.

4. The created rules are listed in the **Filter Rules** view. You can return to editing a selected rule by clicking **Edit**, and you can arrange the order of the rules with the up and down arrows. Place the most specific rules first. In this example, the order is not significant.



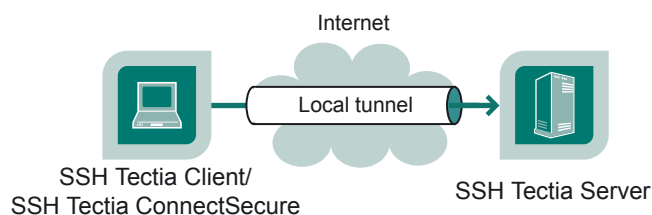
**Figure 7.6. Defining filter rules**

Once transparent TCP tunneling has been activated, it captures the e-mail traffic to the hosts defined in the SSH Tectia configuration and uses encrypted tunnels for sending the data. No changes are required to the configuration of the e-mail application.

When the tunnel is opened, the user is prompted to authenticate to the gateway server. Setting up public-key authentication to the server is recommended. For instructions, see [Section A.1.4.1](#).

### 7.1.2 Non-Transparent TCP Tunneling

When non-transparent TCP tunneling is used, the application to be tunneled is set to connect to the local listener port instead of connecting to the server directly. SSH Tectia Client forwards the connection securely to the remote server.

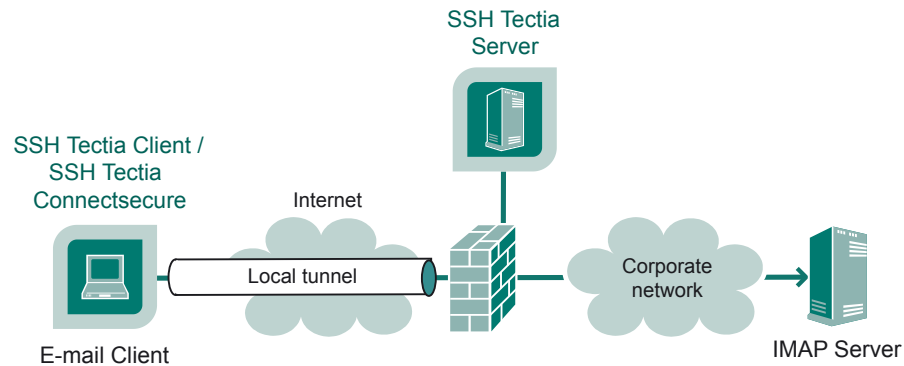


**Figure 7.7. Simple local tunnel**

If you have three hosts, for example, `sshclient`, `sshserver`, and `imapserver`, and you forward the traffic coming to the `sshclient`'s port 143 to the `imapserver`'s port 143, only the connection between the `sshclient` and `sshserver` will be secured. The command you use would be similar to the following one:

```
sshclient$ sshg3 -L 143:imapserver:143 username@sshserver
```

Figure 7.8 shows an example where the Secure Shell server resides in the DMZ network. Connection is encrypted from the Secure Shell client to the Secure Shell server and continues unencrypted in the corporate network to the IMAP server.



**Figure 7.8. Local tunnel to an IMAP server**

Tunnels can also be defined for connection profiles in the Connection Broker configuration file. The defined tunnels are opened automatically when a connection with the profile is made. The following is an example from a `ssh-broker-config.xml` file:

```
<profile id="idl" host="sshserver.example.com">
...
  <tunnels>
    <local-tunnel type="tcp"
      listen-port="143"
      dst-host="imap.example.com"
      dst-port="143"
      allow-relay="no" />
    ...
  </tunnels>
</profile>
```

By default, local tunnels originating only from the client host itself are allowed. To allow also other machines to connect to the tunnel listener port, set the `allow-relay` to `yes`.

The tunneling settings can be made in the SSH Tectia Configuration GUI, under **Connection Profiles** → **Tunneling** per each profile. See [Section A.1.3.8](#).

### 7.1.2.1 Automatic Tunnels

Automatic tunnels are one way of creating non-transparent local tunnels for application connections.

Automatic tunnels always use a connection profile in the tunnel establishing. You can create listeners for local tunnels that will be activated automatically when the Connection Broker starts up. The actual tunnel will be formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.

In the Connection Broker configuration file, make the following kind of settings:

```
<static-tunnels>
  <tunnel type="tcp"
    listen-port="9874"
    dst-host="st.example.com"
    dst-port="9111"
    allow-relay = "no"
    profile="idl" />
</static-tunnels>
```

You can configure the automatic tunnels in the SSH Tectia Configuration GUI, on the **Automatic Tunnels** page. For instructions, see [Section A.1.7](#).

### 7.1.2.2 Examples of Local Tunneling

When **sshg3** is used to create secure tunnels using local port forwarding, the TCP applications to be tunneled are configured to connect to a localhost port instead of the application server port.

Example application, `clientapp1`, by default connects to a Unix server `unix.example.com` using TCP port 2345.

```
$ clientapp1 --username user1 --server unix.example.com --port 2345
```

For securing this TCP application using Secure Shell, use the following commands:

```
$ sshg3 -L 2345:localhost:2345 user1@unix.example.com -S -f &
$ clientapp1 --username user1 --server localhost --port 2345
```

The above **sshg3** command connects to remote Secure Shell server `unix.example.com`, creates a local listener on port 2345, instructs the remote Secure Shell server to forward the incoming traffic to `localhost:2345`, and goes to background in single-shot-mode.

### 7.1.3 Non-Transparent FTP Tunneling

Non-transparent FTP tunneling is an extension to the generic tunneling mechanism. Unlike generic tunneling (port forwarding) mechanism, non-transparent FTP tunneling secures the transferred files, in addition to the FTP control channel. The FTP tunneling code monitors the tunneled FTP control channels and dynamically creates new tunnels for the data channels as they are requested.

When non-transparent FTP tunneling is used, tunnels are created from local client ports to remote servers. The FTP client is configured to connect to SSH Tectia Client which will forward the connection to the endpoint where a Secure Shell server is running.

The typical use case is that SSH Tectia Client is located on the same host as the FTP client; and the FTP server is on the same host as the Secure Shell server. However, other configurations are also supported, but it is worth noticing that the connection is encrypted only between SSH Tectia Client and the Secure Shell server.

Non-transparent FTP tunneling can be requested on the command line, or enabled and defined in the Connection Broker configuration. The configured non-transparent FTP tunneling uses connection profiles, that are defined on SSH Tectia Client.

On command-line, FTP tunneling can be used for both local and remote tunnels. Non-transparent FTP-tunneling is started by entering a `sshg3` command with the following syntax:

```
sshclient$ sshg3 -L ftp/1234:localhost:21 username@sshserver
```

For information on the `sshg3` command, see the [sshg3\(1\)](#) man page.

The FTP tunneling settings can be made in the Connection Broker Configuration GUI, under **Connection Profiles** → **Tunneling** for each profile. See [Section A.1.3.8](#).

FTP tunnels can also be defined for connection profiles in the Connection Broker configuration file. The following is an example from the Connection Broker configuration file `ssh-broker-config.xml`:

```
<profiles>
  <profile id="id1" host="sshserver.example.com"
    ...
    <tunnels>
      <local-tunnel type="FTP"
        listen-port="1234"
        dst-host="127.0.0.1"
        dst-port="21"
        allow-relay="NO" />
      ...
    </tunnels>
  </profile>
</profiles>
```

An FTP connection can then be made with the following (example) commands:

```
sshclient$ ftp
ftp$ open localhost 1234
```

The FTP connection to port 1234 on client is now tunneled to port 21 on the Secure Shell server.

As an alternative to FTP tunneling, you can use the `sftpg3` or `scp3` clients for secure file transfers. These clients can be used on command line or in scripts and they require less configuration than FTP tunneling, since SSH Tectia Server already has `sft-server-g3` as a subsystem, and `sftpg3` and `scp3` clients are included with SSH Tectia Client. Managing remote user restrictions on the server machine will be easier, since you do not have to do it also for FTP.

To understand exactly how FTP tunneling is done, two different cases need to be examined: the active mode and the passive mode of the FTP protocol.

### 7.1.3.1 Tunneling FTP in Passive Mode

In passive mode, the FTP client sends the command `PASV` to the server, which reacts by opening a listener port for the data channel and sending the IP address and port number of the listener as a reply to the client. The reply is of the form `227 Entering Passive Mode (10,1,60,99,6,12)`.

When the Connection Broker notices the reply to the `PASV` command, it will create a local port forwarding to the destination mentioned in the reply. After this the Connection Broker will rewrite the IP address and port in the reply to point to the listener of the newly created local port forwarding (which exists always in a localhost address, 127.0.0.1) and pass the reply to the FTP client. The FTP client will open a data channel based on the reply, effectively tunneling the data through the Secure Shell connection, to the listener the FTP server has opened. The net effect is that the data channel is secured all the way except from the Secure Shell server to the FTP server if they are on different machines. This sequence of events takes place automatically for every data channel.

Since the tunnel is opened to a localhost address, the FTP client must run on the same machine as SSH Tectia Client if passive mode is used.

### 7.1.3.2 Tunneling FTP in Active Mode

In active mode, the FTP client creates a listener on a local port, for a data channel from the FTP server to the FTP client, and requests the channel by sending the IP address and the port number to the FTP server in a command of the following form: `PORT 10,1,60,99,6,12`. The Connection Broker intercepts this command and creates a remote port forwarding from the localhost address of the Secure Shell server to the address and port specified in the `PORT` command.

After creating the tunnel, the Connection Broker rewrites the address and port in the `PORT` command to point to the newly opened remote forwarding on the Secure Shell server and sends it to the FTP server. Now the FTP server will open a data channel to the address and port in the `PORT` command, effectively forwarding the data through the Secure Shell connection. The Connection Broker passes the incoming data to the original listener created by the FTP client. The net effect is that the data channel is secure the whole way except from SSH Tectia Client to the FTP client. This sequence of events takes place automatically for every data channel.

Since the tunnel is made to a localhost address on the SSH Tectia Client machine, the FTP server must be run on the same host as the Secure Shell server if active mode is used.

Where end-to-end encryption of FTP data channels is desired, the FTP server and Secure Shell server need to reside on the same host, and the FTP client and SSH Tectia Client will likewise need to reside on the same host.



## Note

Tunneling FTP in active mode is not guaranteed to work in all setups. If possible, use the passive mode when tunneling FTP connections.

## 7.1.4 SOCKS Tunneling

SOCKS tunneling is a mechanism available for tunneling applications that support the SOCKS4 or SOCKS5 client protocol.

Instead of configuring tunneling (a.k.a port forwarding) from specific ports on the local host to specific ports on the remote server, you can specify a SOCKS server which can be used by the user's applications. Each application is configured in the regular way except that it is configured to use a SOCKS server on a localhost port. The Secure Shell client application, SSH Tectia Client, opens a port in the localhost and mimics a SOCKS4 and SOCKS5 server for any SOCKS client applications.

When the applications connect to services such as IMAP4, POP3, SMTP, and HTTP, they provide the necessary information to the SOCKS server, which is actually SSH Tectia Client mimicking a SOCKS server. SSH Tectia Client will use this information in creating a tunnel to the Secure Shell server and relaying the traffic back and forth securely.

With `sshg3` on the command line, the syntax of the SOCKS tunneling command is as follows:

```
client$ sshg3 -L socks/[listen-address:]listen-port username@sshserver
```

where:

- `[listen-address:]` defines which interface on the client will be listened to (optional argument)
- `listen-port` is the number of the port on the client
- `sshserver` is the IP address or the host name of the Secure Shell server.

For example, the following command will set up a local tunnel from port 1234 on the client to `sshserver`. The applications are set to use a SOCKS server at port 1234 on the client. From the server, the connections are forwarded unsecured to the destination hosts requested by the applications.

```
sshclient$ sshg3 -L socks/1234 username@sshserver
```

SOCKS tunnels can also be defined for connection profiles in the Connection Broker configuration file. The following is an example from a `ssh-broker-config.xml` file:

```
<profile id="idl" host="sshserver.example.com">
...
  <tunnels>
    <local-tunnel type="socks"
      listen-port="1234"
      allow-relay="no" />
  </tunnels>
</profile>
```

```
...  
</tunnels>  
</profile>
```

## 7.2 Remote Tunnels

A remote (incoming) tunnel forwards traffic coming to a remote port to a specified local port.

With `sshg3` on the command line, the syntax of the remote tunneling command is as follows:

```
client$ sshg3 -R [protocol/][listen-address:]listen-port:dst-host:dst-port \  
username@sshserver
```

where:

- `[protocol/]` specifies which protocol is to be used in the tunneled connection, it can be `ftp` or `tcp` (optional argument). The default is `tcp`.
- `[listen-address:]` defines which interface on the remote server will be listened to (optional argument). By default all interfaces are listened.
- `listen-port` is the number of the port on the remote server, and connections coming to this port will be tunneled to the client.
- `dst-host:dst-port` define the destination host address and the port to which the connection is tunneled from the client.
- `sshserver` is the IP address or the host name of the Secure Shell server.

Setting up remote tunneling allocates a listener port on the remote server. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified destination host and port. The connection from the client onwards will not be secure, it is a normal TCP connection.

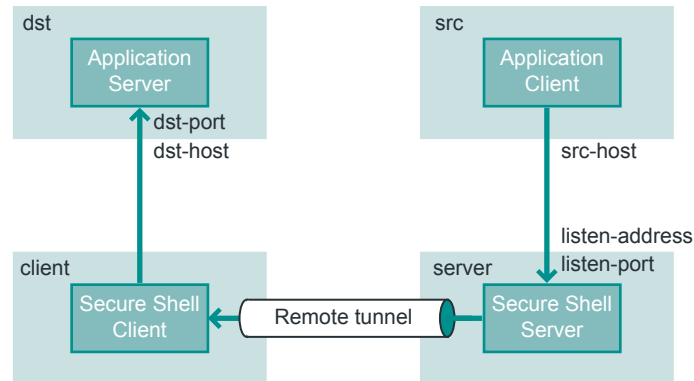


### Note

Every user with access to the remote server host will be able to use remote tunnel.

[Figure 7.9](#) shows the different hosts and ports involved in remote port forwarding.



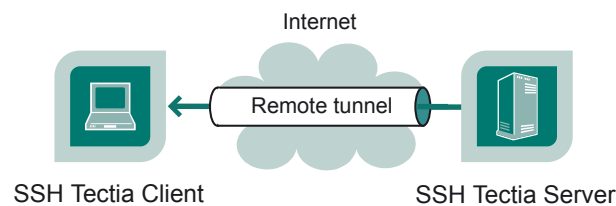


**Figure 7.9. Remote tunneling terminology**

For example, if you issue the following command, all traffic which comes to port 1234 on the server will be tunneled to port 23 on the client. See [Figure 7.10](#).

```
sshclient$ sshg3 -R 1234:localhost:23 username@sshserver
```

The forwarding address in the command is resolved at the (local) end point of the tunnel. In this case `localhost` refers to the client host.



**Figure 7.10. Remote tunnel**

Tunnels can also be defined for connection profiles in the Connection Broker configuration file. The defined tunnels are opened automatically when a connection with the profile is made.

The following is an example from a `ssh-broker-config.xml` file:

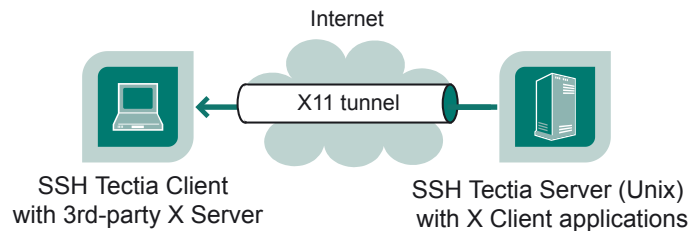
```
<profile id="id1" host="sshserver.example.com">
  ...
  <tunnels>
    <remote-tunnel type="tcp"
      listen-port="1234"
      dst-host="localhost"
      dst-port="23" />
    ...
  </tunnels>
</profile>
```

The tunneling settings can be made in the SSH Tectia Configuration GUI, under **Connection Profiles** → **Tunneling** per each profile. See [Section A.1.3.8](#).

## 7.3 X11 Forwarding

X11 forwarding is a special case of remote tunneling.

SSH Tectia Client supports X11 forwarding on both Unix and Windows platforms. On Windows, you need also the XWindow Manager package. SSH Tectia Server supports X11 forwarding only on Unix platforms.



**Figure 7.11. X11 forwarding**

X11 forwarding can be enabled in the client by setting the following line in the `ssh-broker-config.xml` file (either under `default-settings` or under a connection profile):

```
<forwards>
  <forward type="X11" state="on"/>
</forwards>
```

By default, X11 forwarding is off.

X11 forwarding can be enabled in the SSH Tectia Configuration GUI, under **Default Connection** → **Tunneling** for the default connection, and under **Connection Profiles** → **Tunneling** per each profile. See [Section A.1.2.1.5](#) and [Section A.1.3.8](#).

To test that X11 forwarding works on Windows, use the XWindow Manager. Log into the remote system and type `xclock &`. This starts an X clock program that can be used for testing the forwarding connection.

If the X clock window is displayed properly, you have the X11 forwarding working. If the X clock fails and complains that it cannot open the display, check that the XAuth is properly installed on the remote host.



### Note

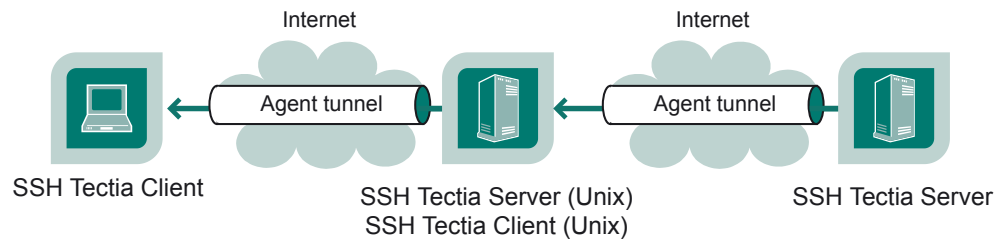
Do *not* set the `DISPLAY` variable on the client. You will most likely disable encryption. (X connections tunneled through Secure Shell use a special local display setting.)

## 7.4 Agent Forwarding

Agent forwarding is a special case of remote tunneling. In agent forwarding, Secure Shell connections and public-key authentication data are forwarded from one server to another without the user having to authenticate

separately for each server. Authentication data does not have to be stored on any other machine than the local machine, and authentication passphrases or private keys never go over the network.

SSH Tectia Client provides authentication agent functionality and the Connection Broker can also serve OpenSSH clients as an authentication agent. SSH Tectia Server supports agent forwarding on Unix platforms. Thus, the start and end points of the agent forwarding chain can be Windows or Unix hosts, but all hosts in the middle of the forwarding chain must be Unix hosts and must have both the Secure Shell client and server components installed.



**Figure 7.12. Agent forwarding**

Agent forwarding can be enabled on the client side both in the default settings and separately for each connection profile.

In the `ssh-broker-config.xml` file, agent forwarding is enabled by setting the following line either under `default-settings` or under a connection profile:

```
<forwards>
  <forward type="agent" state="on" />
</forwards>
```

By default, agent forwarding is disabled (off).

On Windows, agent forwarding can be enabled in the SSH Tectia Configuration GUI, under **Default Connection** → **Tunneling** for the default connection, and under **Connection Profiles** → **Tunneling** per each profile. See [Section A.1.2.1.5](#) and [Section A.1.3.8](#).



## Chapter 8 Troubleshooting SSH Tectia Client

You can run SSH Tectia Client in debug mode to gather information that is useful when troubleshooting any connection or authentication problems.

Do not leave SSH Tectia Client running in debug mode unnecessarily. Debugging slows down the performance.

See [Section 1.2](#) for information on accessing the SSH Tectia online support resources and contacting SSH Tectia Technical Support.

Before you contact SSH Tectia Technical Support, run the **ssh-troubleshoot** (**ssh-troubleshoot.cmd** on Windows) tool to collect necessary information on your system and the installed SSH Tectia products. This information will help in analysing the reported problems, as the Technical Support gets to know exact details about the environment where the SSH Tectia products are running. See instructions in [Section 8.2](#).

### 8.1 Starting Connection Broker in Debug Mode

The Connection Broker is a component included in SSH Tectia Client. The Connection Broker handles all cryptographic operations and authentication-related tasks for SSH Tectia Client and the command-line tools **sshg3**, **scpg3**, and **sftpg3**.

To start the Connection Broker in debug mode, follow these instructions:

1. Open a shell (on Unix) or command prompt window (on Windows).
2. Stop the Connection Broker, if it is currently running. Enter the following command to exit the Connection Broker. This will close all currently open connections of the current user:

```
$ ssh-broker-g3 --exit
```

3. Start the Connection Broker in debug mode by running the following command:

```
$ ssh-broker-g3 -D<filter> -l <logfile>
```

In the command:

- `logfile` specifies the file to which the debug output will be directed
- `filter` is an expression that takes the following syntax: "module=level,module=level,..."
- `module` is an optional expression that can be used to restrict the debug output to only a particular module or to allow the use of varying debug levels for different modules.
- `level` is an integer from 0 (no debug info) to 99 that specifies the desired amount of debug information.

Note that levels 1-9 are the recommended ones. The higher the number, the more detailed the troubleshooting output will be, and the more the debugging will affect performance.

The following example command starts the Connection Broker with global debug level 4 and outputs the debug information to a log file named `broker.log`:

```
$ ssh-broker-g3 -D4 -l broker.log
```

The following example command starts the Connection Broker with debug level 5 for modules starting with "SecShAuth" and level 2 for everything else:

```
$ ssh-broker-g3 -D"SecShAuth*=5,2" -l broker.log
```

4. Connect to a server using one of the clients:

```
$ sshg3 user@host
```

5. View the debug information for the connection in the `broker.log` file.

On Windows, you can view the debug output also in the **Logs** view in the **SSH Tectia Status** window. To open the **SSH Tectia Status** window, right-click the **SSH Tectia** tray icon and select **Status**.

On Unix, you can display the debug output also by using the command line tools with argument `-D`. For example, the following command will display the debug output with a debug level 5 for modules starting with `SecShAuth` and level 2 for modules starting with `sft`:

```
$ sftpg3 -D"SecShAuth*=5,Sft*=2" user@host
```

## 8.2 Collecting System Information for Troubleshooting

SSH Tectia Client includes a troubleshooting tool that automatically collects necessary data about the operating system and hardware, and about the installed SSH Tectia product versions and their configurations into a file. The troubleshooting tool gathers the following information about the system configuration:

- The operating system (OS) version and patches installed
- OS configuration files and other OS information, for example about PAM, syslog, resolver, and ifconfig

- Hardware information, for example the machine model, security class, and CPU version
- OS status, for example the reserved ports and connections per socket
- SSH Tectia binaries, the tool checks the actual installation package versions and detects also debug packages
- SSH Tectia global configuration from the `/etc/` and `/opt/` directories on Unix, and from the `"C:\Program Files\SSH Communications Security\SSH Tectia` directory on Windows
- User-specific SSH Tectia configuration from user's home directory: `$HOME/.ssh2` on Unix, and `"C:\Documents and Settings\<username>"` or `"C:\Users\"` on Windows
- The user account running the troubleshooting tool
- On Unix, it is configurable if everything stored in the specified user's configuration directories, including the private keys, are to be collected. This helps the Technical Support to better simulate the user's situation.

On Unix, run the troubleshooting tool with command:

```
# ssh-troubleshoot [options] info [command-options]
```

On Windows, run the troubleshooting tool with command:

```
ssh-troubleshoot.cmd [options] info
```

For details about the command options, refer to [ssh-troubleshoot\(1\)](#).

The collected data is stored in the results file named as follows:

- On Unix: `ssh-troubleshoot-data-<hostname>-<timestamp>.tar`
- On Windows: `ssh-troubleshoot-data-<hostname>-<timestamp>.log`

In the file name, `hostname` identifies the host from where the information was collected, and `timestamp` specifies the date and time when the information was stored into the file. The timestamp format is `yyyymmdd-hhmmUTC`. So the reports are not in local time, but use the UTC.

You can send the file to SSH Tectia Technical Support for analysis.



## Note

Handle the output file with appropriate care as it may contain security-critical data.

## 8.3 Answers to Common Problems

This sections introduces workaround instructions for some problem situations.

### Troubleshooting GSSAPI Authentication

When connecting from a Windows 5.x or 6.x client to a Windows 4.x server using GSSAPI authentication, if authentication fails although GSSAPI has been correctly configured, you may have to disable the LMHOSTS lookup on the client-side computer. Follow these instructions:

1. Select **Control Panel** → **Network Connections**.
2. In **Local Area Connection**, right-click and select **Properties**.
3. In the **Local Area Connection Properties** dialog box, **General** tab, select **Internet Protocol (TCP/IP)** and click the **Properties** button.
4. In the **Internet Protocol (TCP/IP) Properties** dialog box, in the **General** tab, click the **Advanced** button.
5. In the **Advanced TCP/IP Settings** dialog box, in the **WINS** tab, clear the **Enable LMHOSTS lookup** check box.
6. Restart the client-side computer.

### Password Window Loses Focus

In some cases on Windows XP clients, the window for the password or the public-key passphrase can lose focus. This means that the password window is shown active on the screen, but when you start typing your password, the asterisks do not appear in the box. If the actual focus happens to be in another window, your password or passphrase may appear there.

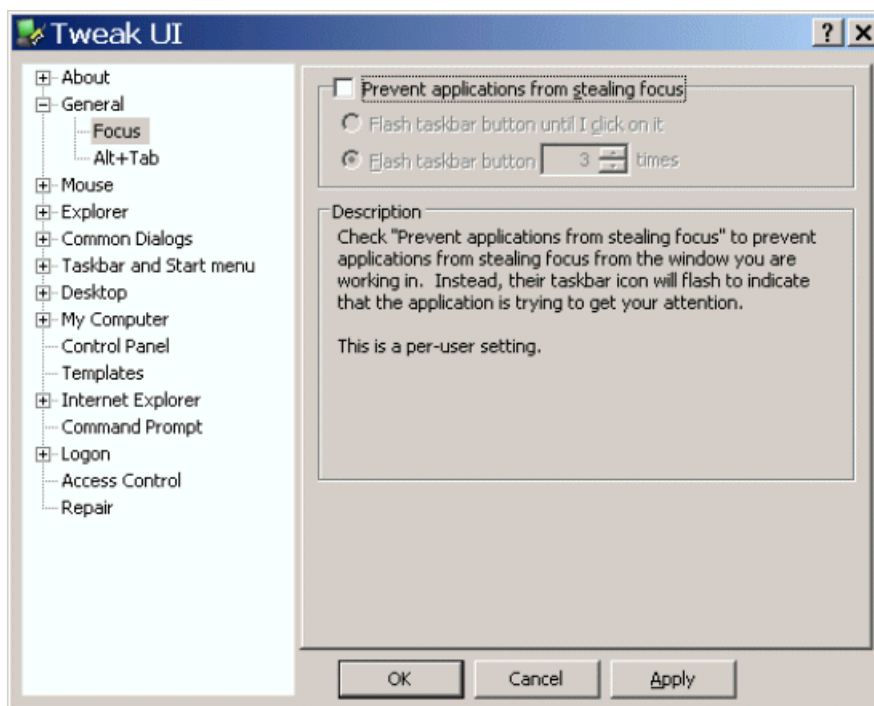
The password window loses focus most often when you have the SSH Tectia Connection Broker status window open. So the first workaround is to make sure the status window is not open while you start logging in.

A permanent solution to the problem is to modify the Windows settings that control the behaviour of the applications. By default, Windows has on a setting that prevents applications from stealing the focus. As a workaround, you can allow applications to steal the focus. Note that the setting affects all Windows applications, not just the SSH Tectia Connection Broker.

To be able to change the setting, you need to download the Microsoft Tweak UI utility program. Follow these instructions:

1. Download the Microsoft Tweak UI utility from: <http://www.microsoft.com/windowsxp/downloads/power toys/xppowertoys.mspx>
2. Install the Tweak UI on your computer with the help of the installation wizard included.
3. Start the program from the Start menu.
4. Go to the **General - Focus** view and clear the **Prevent applications from stealing focus** check box as shown in the following figure.





**Figure 8.1. Microsoft Tweak UI utility**

5. Click **Apply** or **OK**.



---

# Appendix A Connection Broker Configuration Tools

The Connection Broker is a component included in SSH Tectia Client, and it handles all cryptographic operations and authentication-related tasks for SSH Tectia Client. For this reason, all authentication and connection profile settings are made in the Connection Broker configuration.

The Connection Broker configuration can be edited and viewed via the SSH Tectia Configuration GUI. For instructions, see [Section A.1](#).

The Connection Broker stores the settings in an XML-based configuration file `ssh-broker-config.xml`. It is possible to edit the configuration file directly with an ASCII-text editor or an XML editor. For description of the configuration file, see [ssh-broker-config\(5\)](#).


## A.1 SSH Tectia Connections Configuration GUI


You can use the SSH Tectia Connections Configuration user interface to edit the authentication and connection profile settings on the Connection Broker included in SSH Tectia Client.


The SSH Tectia Connections Configuration interface functions are mostly the same on all supported platforms, but the GUI looks vary slightly depending on the platform-specific settings. In this document, we show the Windows GUI in detail, and specify the differences in the Unix platform GUIs.

### A.1.1 Opening the GUI

On Windows, the SSH Tectia Connections Configuration interface can be accessed in several ways:

- Right-click on the SSH Tectia tray icon  in the Windows taskbar notification area to access the shortcut menu, and select **Configuration**.
- On the Windows **Start** menu, select **Start** → **Programs** → **SSH Tectia Client** → **SSH Tectia Connections Configuration**.

- When you have the SSH Tectia Client GUI active, click the SSH Tectia Connections Configuration tool icon  in the toolbar or select **Edit** → **SSH Tectia Connections Configuration** in the menu bar.

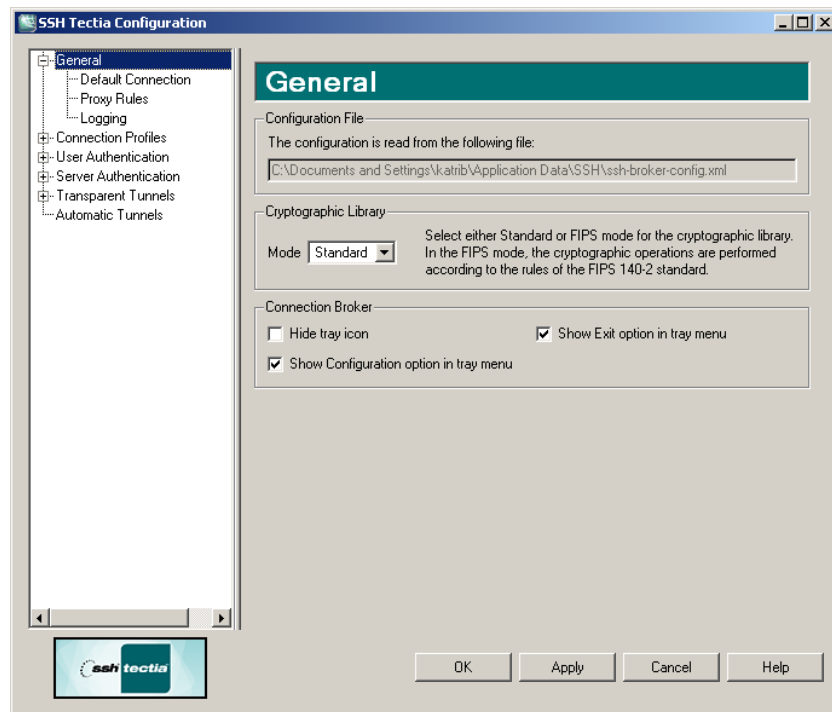
On Windows, SSH Tectia Client includes a separate GUI for configuring the user interface settings for the SSH Tectia terminal GUI. Open the **User Interface Settings** tool by clicking the  icon in the toolbar of the SSH Tectia terminal GUI. For instructions on the GUI configurations, see [Appendix B](#).

## Note

If the Connection Broker configuration is generated and controlled by SSH Tectia Manager, the GUI hides those views that contain Manager-configured settings. This protects the configuration from unauthorized local modifications.

### A.1.2 Defining General Settings

On the **General** page, you can select the cryptographic library to be used and define the SSH Tectia tray icon settings.



**Figure A.1. General settings**

#### Configuration File

Shows the location of the user-specific Broker configuration file. The default location is "%APP-DATA%\SSH\ssh-broker-config.xml".

Each time the configuration file is saved, a backup of the old configuration is stored in "%APP-DATA%\SSH\ssh-broker-config-backup.xml".

### Cryptographic Library

SSH Tectia Client can be operated in *FIPS mode*, using a version of the cryptographic library that has been validated according to the Federal Information Processing Standard (FIPS) 140-2. In this mode, the cryptographic operations are performed according to the rules of the FIPS 140-2 standard.

Select whether to use the **Standard** or the **FIPS 140-2** certified version of the cryptographic library.



### Note

Setting the FIPS mode does not prevent using algorithms from crypto plugins. For example, CryptiCore can be used even when the main crypto library is set in the FIPS mode. To enforce that only FIPS-compliant algorithms are used, disable the non-FIPS algorithms from the configuration. For the default settings, see [Section A.1.2.1.2](#), [Section A.1.2.1.3](#), and for the profile-specific settings, see [Section A.1.3.4](#), and [Section A.1.3.5](#).

### Connection Broker

Select whether to hide the SSH Tectia tray icon from the Windows task bar, and whether to show the **Exit** and **Configuration** options in the shortcut menu.

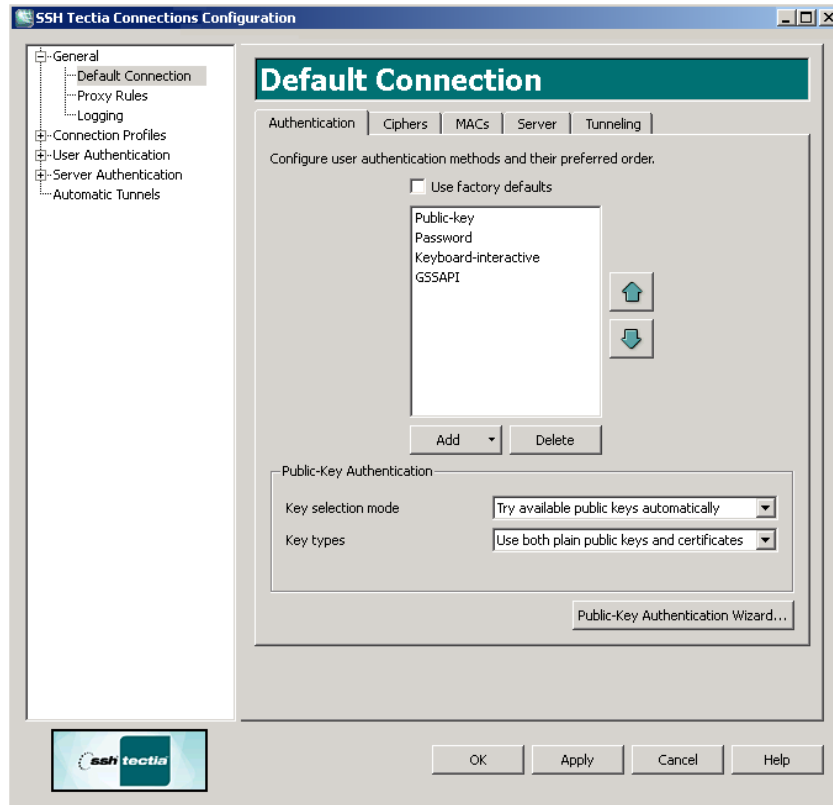
## A.1.2.1 Defining Default Connection Settings

The **Default Connection** page allows you to edit the default settings for authentication ([Section A.1.2.1.1](#)), ciphers ([Section A.1.2.1.2](#)), MACs ([Section A.1.2.1.3](#)), server connections ([Section A.1.2.1.4](#)), and tunneling ([Section A.1.2.1.5](#)), and

Newly created connection profiles will inherit the default settings defined here. The values can be customized on the profile-specific tabbed pages and they override the default settings. See [Section A.1.3.2](#), [Section A.1.3.4](#), [Section A.1.3.5](#), and [Section A.1.3.6](#).

### A.1.2.1.1 Defining Authentication

On the **Authentication** tab, you can define the default user authentication methods.



**Figure A.2. Authentication methods for SSH Tectia Client**

Select the **Use factory defaults** check box to use the factory default authentication methods, or clear the check box to define a custom list of authentication methods.

In SSH Tectia Client 6.1, the factory default authentication methods are, in order:

- Public-key
- Password
- Keyboard-interactive
- GSSAPI

To add a new authentication method to the list, click **Add** and select the method from the drop-down menu.

To remove an authentication method, select the method from the list and click **Delete**.

Use the arrow buttons to organize the preferred order of the authentication methods. The first method that is allowed by the Secure Shell server is used. Note that in some cases, the server may require several authentication methods to be passed before allowing login.

Possible methods for user authentication are:

- **Password:** Users are requested to enter a password for authentication.
- **Public-key:** Users are requested to use public-key authentication. See also [Section A.1.4](#).

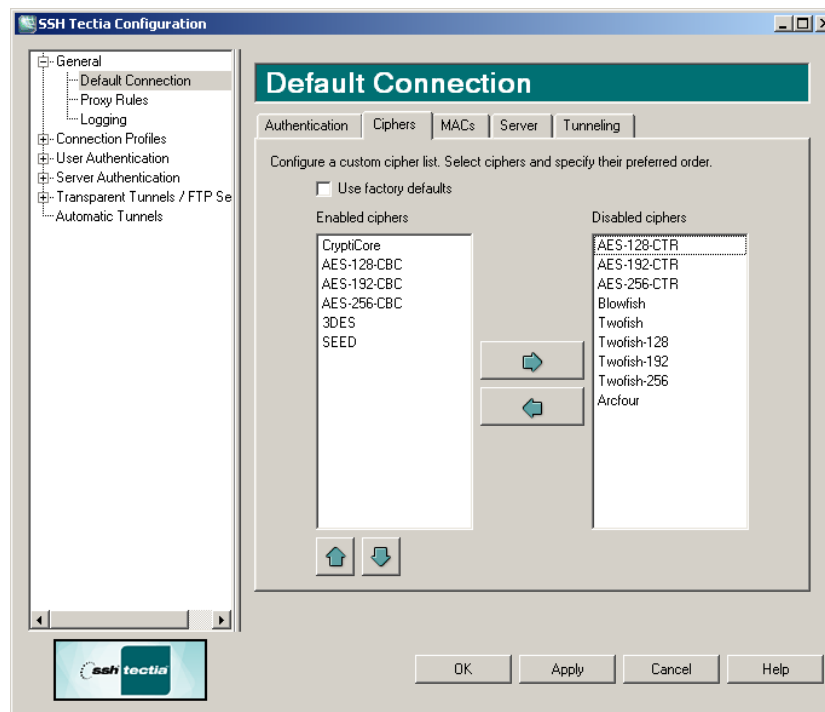
In the **Public-key Authentication** field, with setting **Key selection mode** you can define whether the public key is selected automatically or by the user. With the **Key types** setting you can select whether only certificates or plain public keys or both will be used to authenticate the user. By default, SSH Tectia Client tries all public keys automatically.

To generate new public-key pairs and to upload the public part of the key to a server, click the **Public-Key Authentication Wizard** button.

- **Keyboard-interactive:** Keyboard-interactive is designed to allow the Secure Shell client to support several different types of authentication methods, including RSA SecurID, and PAM. For more information on keyboard-interactive, see [Section 5.7](#).
- **GSSAPI:** GSSAPI (Generic Security Service Application Programming Interface) is a common security service interface that allows different security mechanisms to be used via one interface. For more information on GSSAPI, see [Section 5.8](#).

#### A.1.2.1.2 Defining Ciphers

On the **Ciphers** tab, you can define the encryption algorithms used.



**Figure A.3. Defining a cipher list**

Select the **Use factory defaults** check box to use the factory default algorithms, or define a cipher list using the arrow buttons. The ciphers are tried in the order they are specified.

The factory default ciphers are, in order:

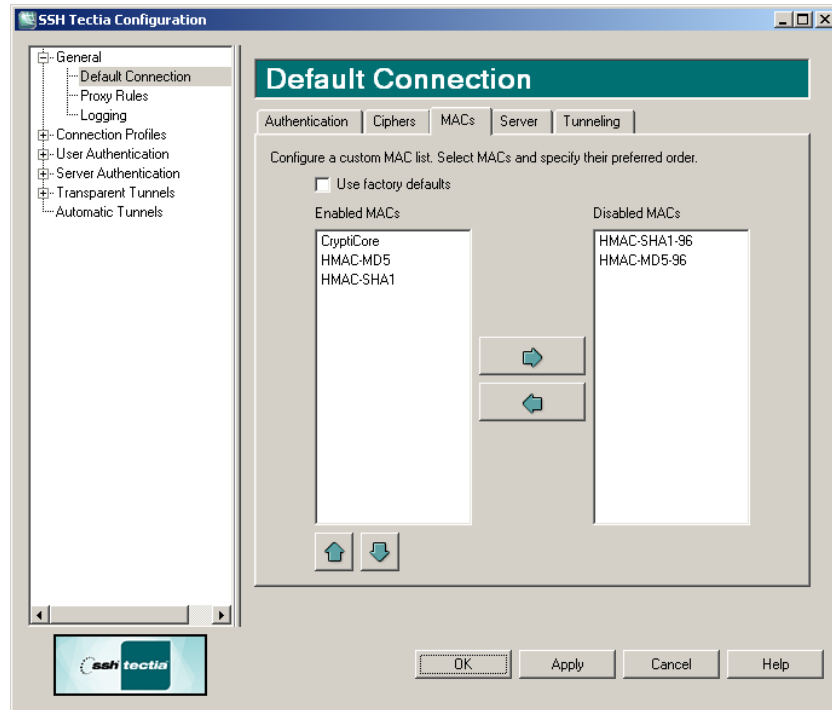
- CryptiCore
- AES-128-CBC
- AES-192-CBC
- AES-256-CBC
- AES-128-CTR
- AES-192-CTR
- AES-256-CTR
- 3DES
- SEED

The ciphers that can operate in the FIPS mode are 3DES and the CBC-mode AES-128, AES-192, and AES-256. (The counter mode AES ciphers are not available in FIPS mode.)

#### **A.1.2.1.3 Defining MACs**

On the **MACs** tab, you can configure the message integrity algorithms used.





**Figure A.4. Defining a MAC list**

Select the **Use factory defaults** check box to use the factory default algorithms, or define a MAC list using the arrow buttons. The MACs are tried in the order they are specified.

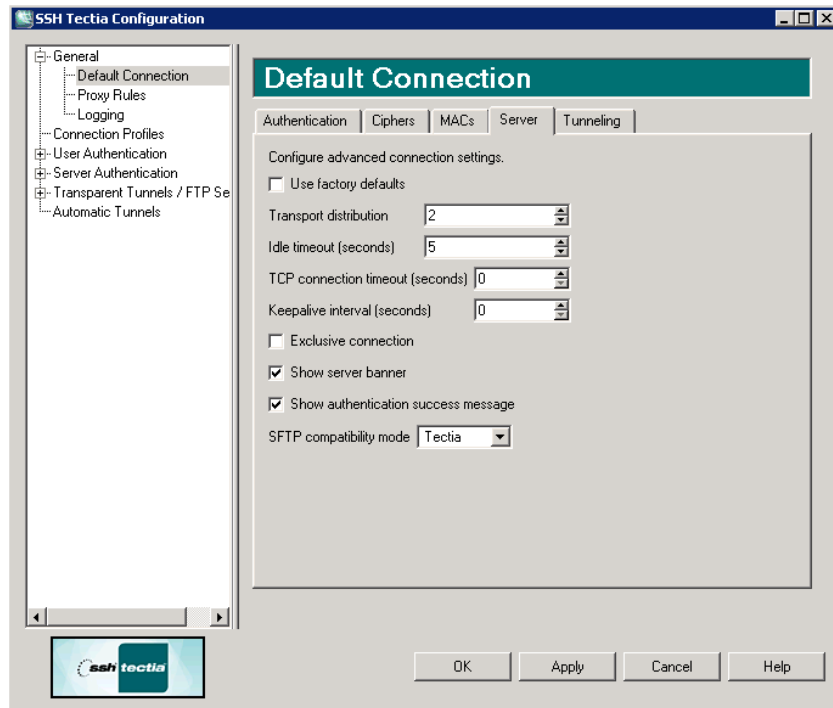
The factory default MACs are, in order:

- CryptiCore
- HMAC-MD5
- HMAC-SHA1

The HMAC-SHA1 algorithm can operate in the FIPS mode.

#### **A.1.2.1.4 Defining Server Connections**

On the **Server** tab, you can define advanced server connection settings.



**Figure A.5. Defining server connection settings**

### Use factory defaults

Select the check box to use the default values for the server connection settings.

### Idle timeout

Specify how long idle time (after all connection channels are closed) is allowed for a connection before automatically closing the connection. The default is 5 seconds. Setting a longer time allows the connection to the server to remain open even after a session (for example, SSH Tectia terminal GUI) is closed. During this time, a new session to the server can be initiated without re-authentication. Setting the time to 0 (zero) terminates the connection immediately when the last channel to the server is closed.

### TCP Connection Timeout

Specify for how long a TCP connection will be attempted to a Secure Shell server. Define the timeout in seconds, and after that time the TCP connection will be released in case the remote server is down or unreachable. Setting the value as 0 (zero) means this SSH Tectia setting is disabled and the system default TCP timeout will be used. By default, the system timeout is used.

### Keepalive interval

Specify an interval (in seconds) for sending keepalive messages to a Secure Shell server. The default is 0, meaning that no keepalive messages are sent.

### Exclusive connection

Select this check box if you want always a new connection opened, instead of reusing a currently open connection.

### Show server banner

Select this check box if you want to have the server banner message file (if it exists) visible to users before login.

### Show authentication success message

Unselect this check box if you do not want to have the `AuthenticationSuccessMsg` messages output and logged. By default the messages are enabled.

### SFTP compatibility mode

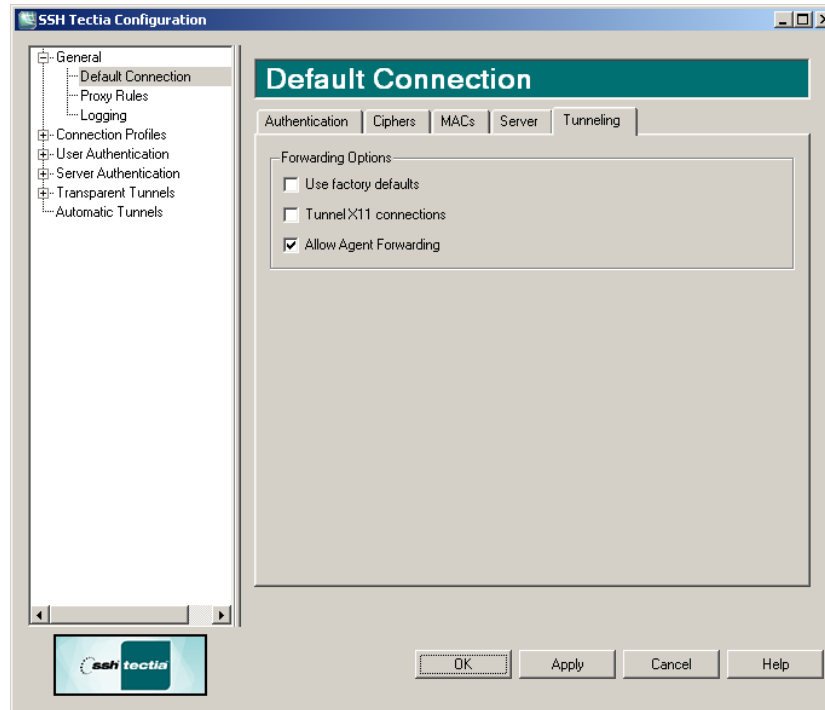
Select a suitable mode for transferring files with SFTP. This setting affects the behaviour of the `get/mget/sget` and `put/mput/sput` commands and the recursion level used by the **sftpg3** client. The following options are available:

- `tectia` (the default) - **sftpg3** transfers files recursively from the current directory and all its subdirectories.
- `ftp` - the `get/put` commands are executed as `sget/sput` meaning that they transfer a single file, and no subdirectories are copied.
- `openssh` - copies only regular files and symbolic links from the specified directory, and no subdirectories are copied. Otherwise the semantics of the ``get'` command is unchanged.

The recursion depth can be overridden by using the **sftpg3** client's commands `get/put/mget/mput` with command-line option `--max-depth="LEVEL"`. For more information, see [sftpg3\(1\)](#).

#### A.1.2.1.5 Defining Default Tunneling Settings

On the **Tunneling** tab, you can define the default settings for X11 connections and agent forwarding (tunneling). The defaults are applied to new connection profiles and to those connection profiles that do not have their own tunneling settings defined.



**Figure A.6. Defining default tunneling settings**

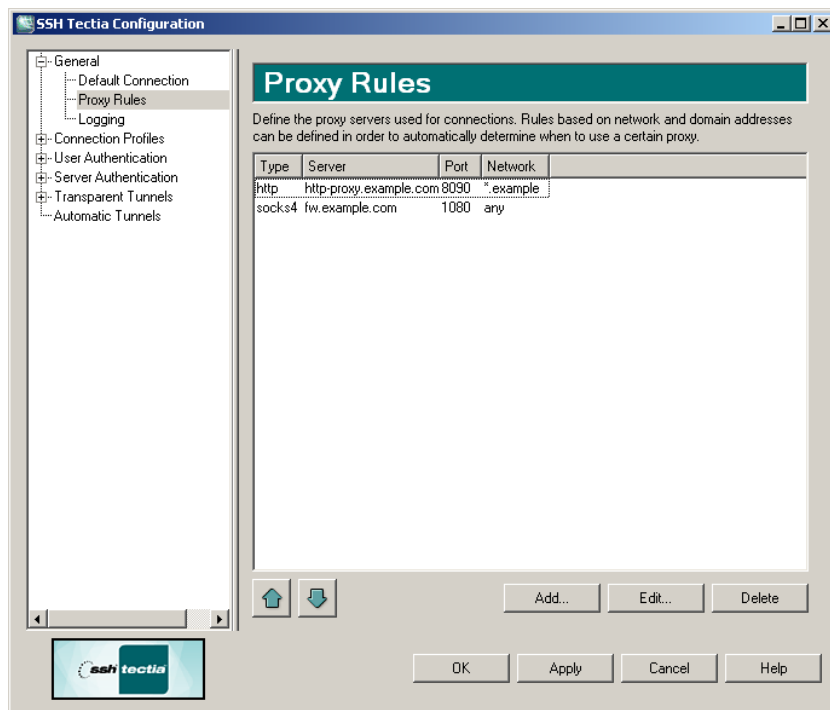
Select the **Use factory defaults** check box to apply the factory defaults for X11 and agent forwarding. According to the factory defaults, both forwarding methods are disabled (off).

Select the **Tunnel X11 connections** check box to allow X11 forwarding on the client side.

Select the **Allow Agent Forwarding** check box to allow agent forwarding on the client side.

### **A.1.2.2 Defining Proxy Rules**

On the **Proxy Rules** page, you can define proxy rules to be used for connections.



**Figure A.7. Defining proxy rules**

To add a new proxy rule:

1. Click **Add**. The **Proxy Rule** dialog box opens.
2. Select the **Type** of the rule. The type can be **Direct** (no proxy), **Socks4**, **Socks5**, or **Http**.



**Figure A.8. Defining proxy settings**

For other types than direct, enter the proxy **Server** address and **Port**.

Select also whether the proxy rules applies to **Any** connection or only to connections to the specified **Network**. In the **Network** field, you can enter one or more conditions delimited by commas (,). The conditions can specify IP addresses or DNS names.

The IP address/port conditions have an address pattern and an optional port range (`ip_pattern[:port_range]`).

The `ip_pattern` may have one of the following forms:

- a single IP address `x.x.x.x`
- an IP address range of the form `x.x.x.x-y.y.y.y`
- an IP sub-network mask of the form `x.x.x.x/y`

The DNS name conditions consist of a hostname which may be a regular expression containing the characters "\*" and "?" and a port range (`name_pattern[:port_range]`).

Click **OK**.

To edit a proxy rule, select a rule from the list and click **Edit**.

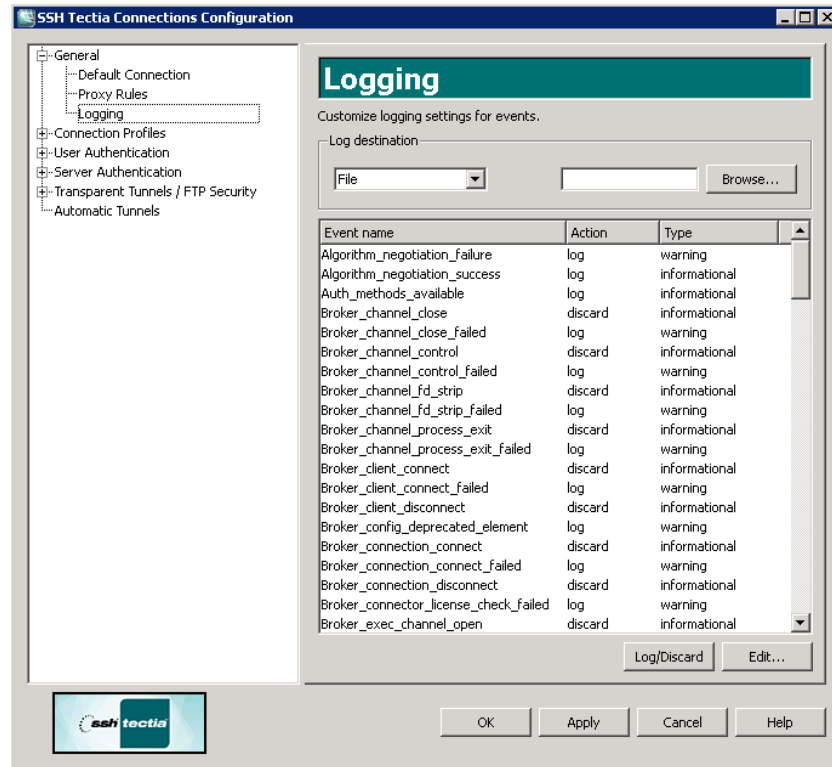
To delete a proxy rule, select a rule from the list and click **Delete**.

The rules are read from top down. Use the arrow button to change the order of the rules.

To use these general proxy rules with a connection profile, you must select to do so in the profile settings. See [Section A.1.3.7](#).

### A.1.2.3 Defining Logging Settings

On the **Logging** page, you can enable logging and customize the information that will be logged in the event log. By default logging is disabled.



**Figure A.9. Logging settings**

To enable logging of SSH Tectia Client internal events, select how the logs will be saved. In the **Log Destination** field, select **File** to have the log data saved in to a file named in the field on the right. Enter the exact file name or browse to an existing file. To have the SSH Tectia Client data stored in the hosts Event Log, select **Event Log**.

Each event has an associated **Action** and **Type**. They have reasonable default values, which are used if no explicit logging settings are made.

The action can be either **log** or **discard**.

The event type can be one of the following:

- **Informational**
- **Warning**
- **Error**
- **Security success**
- **Security failure**

For a description of the log events, see [Appendix E](#).

To change whether the event is logged or not, select an event from the list and click **Log/Discard**. You can select multiple events by holding down the **SHIFT** or **CTRL** key while clicking.

To customize the event action and type, select an event from the list and click **Edit**. You can select multiple events by holding down the **SHIFT** or **CTRL** key while clicking. The **Edit Audit** dialog box opens. Select the **Action** (log or discard) and the **Type** (informational, warning, error, security-success or security-failure) for the event and click **OK**.

### A.1.3 Defining Connection Profiles

Under **Connection Profiles** you can configure separate connection settings for each Secure Shell server you connect to. You can also configure several profiles for the same server, for example, with different user accounts.

To add a connection profile, click **Add profile** in the **Connection Profiles** page. Enter a name for the profile and click **OK**. By default, the profile name is also used as the hostname of the server.

Newly created connection profiles will inherit the default values for authentication, ciphers, MACs, and advanced server settings defined under the **General** → **Defaults** page ([Section A.1.2.1](#)). The values can be customized on the profile-specific tabbed pages.

When a profile is added from the SSH Tectia GUI using the **Add Profile** option, the initial window type of the new profile is automatically set to be the same as in the current GUI view.

Define the profile settings in the tabbed view as described in [Section A.1.3.1](#), [Section A.1.3.2](#), [Section A.1.3.4](#), [Section A.1.3.5](#), [Section A.1.3.6](#), [Section A.1.3.7](#), [Section A.1.3.8](#), [Section A.1.3.9](#), [Section A.1.3.10](#), [Section A.1.3.11](#), [Section A.1.3.12](#), and [Section A.1.3.13](#).

If you have a lot of different servers you are connecting to, you can organize the connection profiles in folders. To add a folder for connection profiles, click **Add folder** in the **Connection Profiles** page. Enter a name for the folder and click **OK**. You can now add connection profiles to the folder by selecting the folder and clicking **Add profile**. The profile will be created in the folder.

To move a profile to a different profile folder, select the profile from the list and click **Move**. Select the folder where you want to move the profile from the drop-down list and click **OK**.

To rename a connection profile or a profile folder, select a profile or a folder and click **Rename**. Enter a new name and click **OK**.

To remove a connection profile or a profile folder, select a profile or a folder and click **Delete**. You will be asked for confirmation. Click **OK** to proceed with the deletion.

Note that removing a profile folder removes also all profiles in it.

Click **Test Connection** to open a connection to the remote server. You need to connect to the server once in order to get the server's host key. SSH Tectia Client will prompt you to verify the received key. Check that

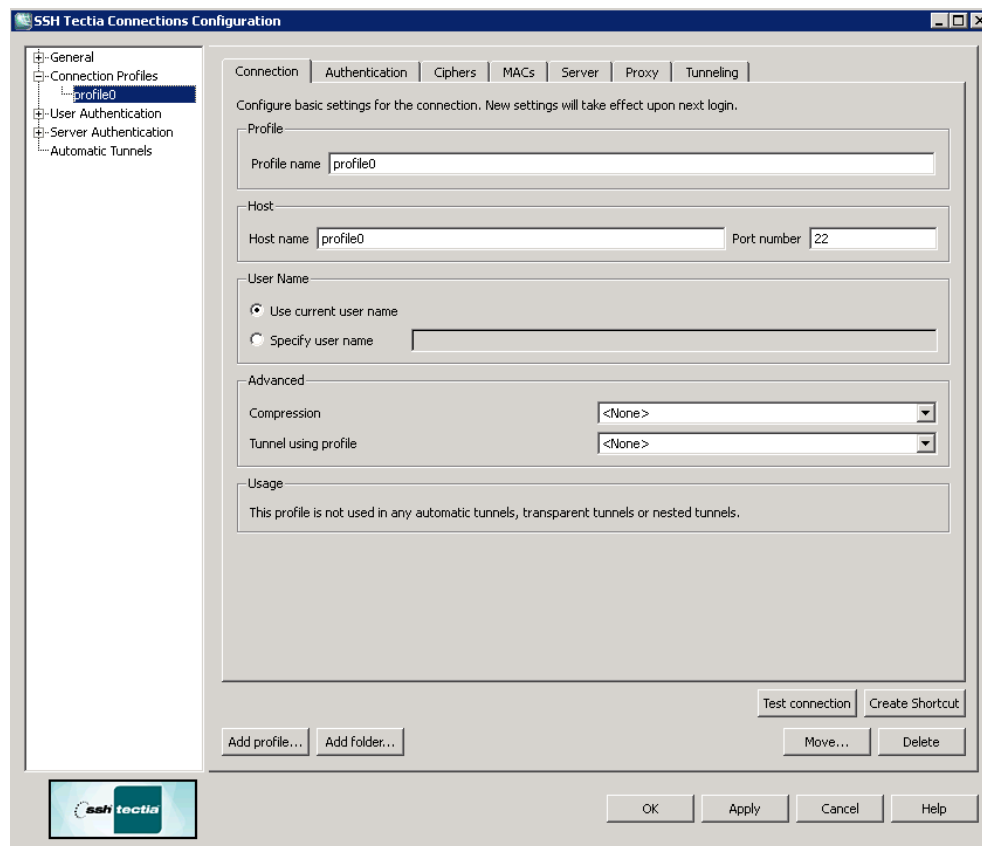


it is valid, preferably by calling the server's administrator, and save the validated key. After this, the locally saved information on the key will be used in the authentication process automatically.

To add a shortcut to the created profile on your desktop on Windows, click **Create Shortcut**. When you double-click the icon, it will directly open a connection to the host defined in the profile.

### A.1.3.1 Defining Connection Settings

On the **Connection** tab, you can define the protocol settings used in the connection. Any changed connection settings will take effect the next time you log in.



**Figure A.10. Configuring connection profiles**

#### Host Name

Specify the host name or the IP address of the remote host computer to which you want to connect with the profile.

When transparent TCP tunneling is used on Windows, '%DESTINATION\_HOSTNAME%' is supported as the hostname definition. This option exists for backward compatibility reasons. From release 6.0 onwards, you can define that SSH Tectia Client uses the destination IP address received from the tunneled application with setting **Transparent Tunnels** → **Filter Rules** → **Use host name from the application** (in XML configuration: `hostname-from-app="yes"`).

### Port Number

Define the listen port on the Secure Shell server. The default SSH port number is 22. In case you know that the remote server uses another port, enter the number in the **Port Number** field.



### Note

A Secure Shell server program must be listening to the specified port on the remote host computer or the connection attempt will not succeed. If you are unsure which port the remote host computer is listening to, contact the system administrator of the remote host.

### User Name

Select **Use current user name** if the connection should always be made using the currently logged in user name. This is similar to defining %USERNAME% (note the percent signs) as the user name. %USERNAME% reads the actual user name from an environment variable.

Select **Specify user name** and enter the user name, if you want to define the user name to be used when connecting to the remote host computer.

### Advanced

In **Compression**, select the desired compression setting from the drop-down menu. Valid choices are **zlib** and **none**. Compression is disabled by default.

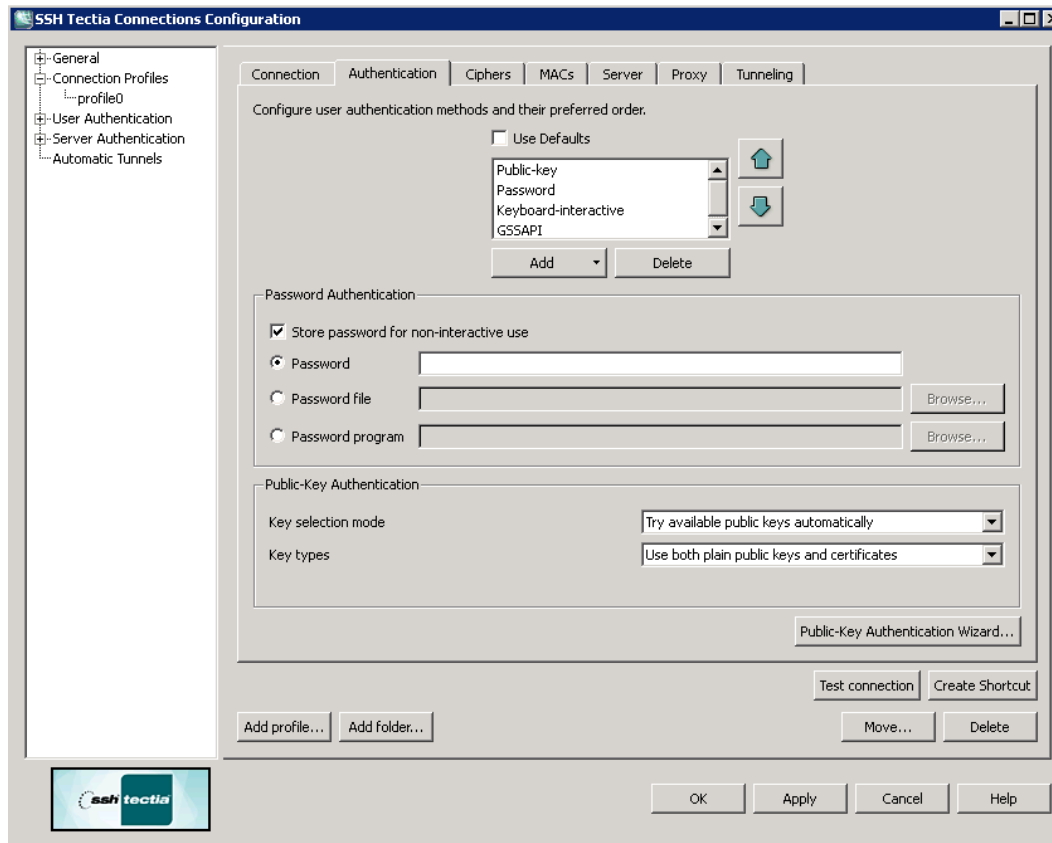
In **Tunnel using profile**, use the drop-down list to select a profile for creating a nested tunnel. The first tunnel will be created to the server defined in the current connection profile, and from there, the second tunnel will be created to a host defined in the profile selected with the **Tunnel using profile** setting. By default, tunneling is disabled.

### Usage

This field shows information on where the defined profile is used.

## A.1.3.2 Defining Authentication

On the **Authentication** tab, you can define the user authentication methods for the profile.



**Figure A.11. Configuring authentication methods for the profile**

1. Select the **Use Defaults** check box to use the authentication methods defined on the **Default Connection** page ([Section A.1.2.1.1](#)), or clear the check box to define a custom list of authentication methods.

To add a new authentication method to the list, click **Add** and select the method from the drop-down menu.

To remove an authentication method, select a method from the list and click **Delete**.

Use the arrow buttons to organize the preferred order of the authentication methods. The first method that is allowed by the Secure Shell server is used. Note that in some cases, the server may require several authentication methods to be passed before allowing login.

Possible methods for user authentication are:

- **Password:** Use a password for authentication.
- **Public-key:** Use public-key authentication. See also [Section A.1.4](#).
- **Keyboard-interactive:** Keyboard-interactive is designed to allow the Secure Shell client to support several different types of authentication methods, including RSA SecurID, and PAM. For more information on keyboard-interactive, see [Section 5.7](#).

- **GSSAPI:** GSSAPI (Generic Security Service Application Programming Interface) is a common security service interface that allows different security mechanisms to be used via one interface. For more information on GSSAPI, see [Section 5.8](#).
2. If you want to use the profile in non-interactive connections, you can select to store a password with the profile in the **Password Authentication** field.

Select **Password** to enter the actual password string.

Select **Password file** to enter a path to a file containing the password.

Select **Password program** to enter a path to a program or a script that outputs the password.



### Caution

If the password is given using this option, it is extremely important that the `ssh-broker-config.xml` file, the password file, or the program are not accessible by anyone else than the intended user.



### Note

Any password given with the command-line options will override this setting.

3. If you want to define specifically how the public-key authentication behaves, clear **Use defaults** on the **Authentication** tab, make sure `Public-key` is included in the list, and choose the options in the **Public-key Authentication** field. You can define whether the public key is selected automatically or by the user, and whether only certificates or plain public keys or both will be used to authenticate the user. By default, SSH Tectia Client tries all public keys automatically.
4. To generate a public-key pair and to upload it to the remote server, click the **Public-Key Authentication Wizard** button. For instructions, see [Section A.1.3.3](#).
5. Click OK to save the connection profile.

#### A.1.3.3 Using the Public-Key Authentication Wizard

On Windows, you can use the SSH Tectia **Public-Key Authentication Wizard** to generate and to upload public-key pairs. The wizard will generate two key files, your private key and your public key.

The new private and public key will be stored on your local computer in the `%APPDATA%\SSH\UserKeys` directory. The private key file has no file extension, and the public key has the same base file name as the private key, but with `.pub` as the file extension.

Select the **Keys and Certificates** page under **User authentication** and click **New Key** to start the Public-Key Authentication Wizard.

**Figure A.12. The Public-Key Authentication Wizard**

Define the key properties and the required passphrase to protect your key pair; you will be requested to enter the passphrase always when using the keys to authenticate yourself.

#### File Name

Type a unique name for the key file. SSH Tectia Client suggest a name consisting of the user name and the host name.

#### Comment

In this field you can write a short comment that describes the key pair. You can for example describe the connection the keys are used for. This field is not obligatory, but helps to identify the key later.

#### Passphrase

Type a phrase that you have to enter when handling the key. This passphrase works in a similar way to a password and gives some protection for your private key.

Make the passphrase difficult to guess. Use at least 8 characters, both letters and numbers. Any punctuation characters can be used as well.

Memorize the passphrase carefully, and do not write it down.

For connections where no user interaction is available, you can consider leaving the password empty.

#### Retype passphrase

Type the passphrase again. This ensures that you have not made a typing error.

Click the **Advanced Options**, to define the type of the key to be generated and the key length to be different from the defaults. By default, SSH Tectia Client generates a pair of 2048-bit DSA keys.

In the **Key Properties** fields, you can make the following selections:

#### Key Type

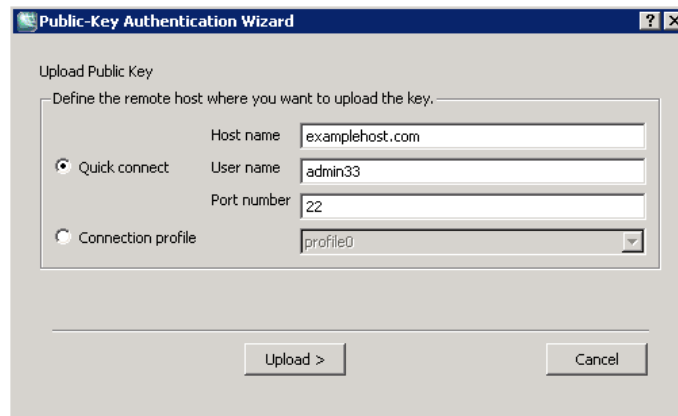
Select the type of the key to be generated. Available options are DSA or RSA.

#### Key Length

Select the length (complexity) of the key to be generated. Available options are 1024, 2048 or 3072 bits.

Larger keys are more secure, but slower to generate.

As soon as a new key has been generated, the Wizard proceeds to uploading the key to a remote server. In case you want to upload an existing key to a remote server, select the key file in the Keys and Certificates view, and click **Upload**. The following dialog appears in both cases:



**Figure A.13. Uploading a key**

In the **Upload Public Key** view of the wizard, define the remote host where to upload the key:

#### Quick connect

Select this option to define the remote **Host name** and your **user name** there. The default Secure Shell port is 22.

#### Connection profile

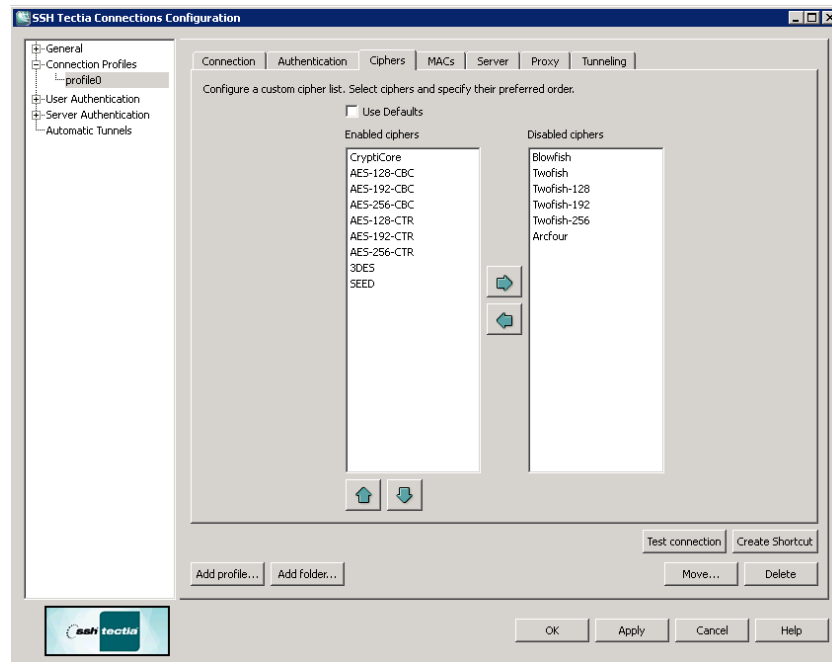
Select a **Connection profile** from the drop-down list that specifies the desired remote host and user name.

Click **Upload** to upload the key to the selected server. If you are already connected to the remote server host, the key upload starts immediately. If you are not connected, you will be prompted to authenticate on the server (by default with password).

The public key will be uploaded to the default user home directory (%USERPROFILE%\ .ssh2 on Windows, \$HOME/ .ssh2 on Unix).

### A.1.3.4 Defining Ciphers

On the **Ciphers** tab, you can define the encryption algorithms used for the profile.

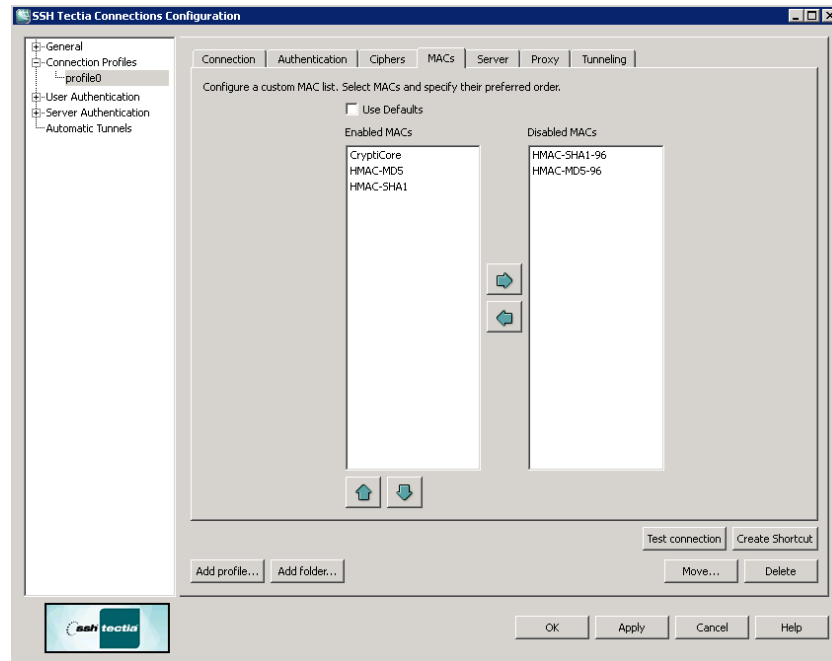


**Figure A.14. Defining a cipher list for the profile**

Select the **Use Defaults** check box to use the algorithms defined on the **Default Connection** page ([Section A.1.2.1.2](#)), or define a cipher list using the arrow buttons. The ciphers are tried in the order they are specified.

### A.1.3.5 Defining MACs

On the **MACs** tab, you can configure the message integrity algorithms used for the profile.



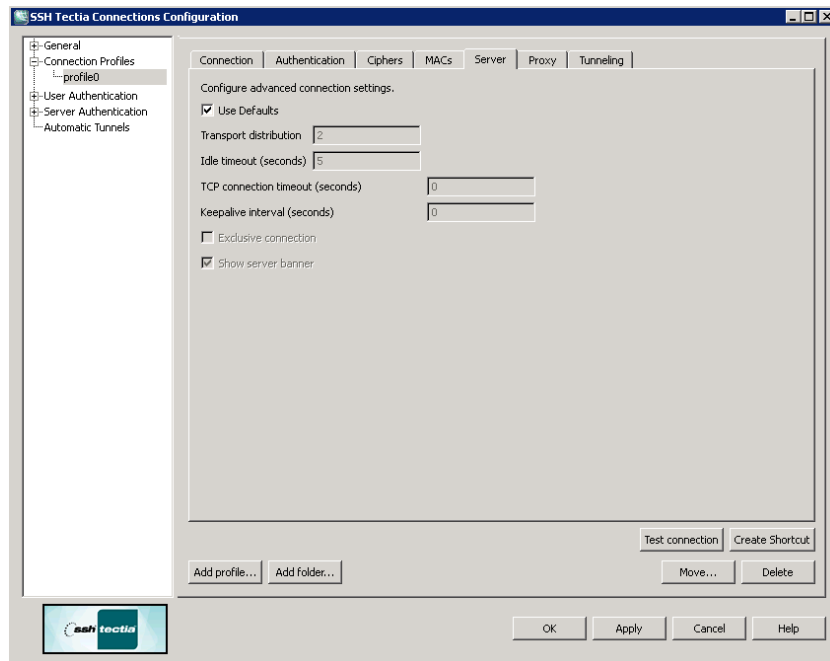
**Figure A.15. Defining a MAC list for the profile**

Select the **Use Defaults** check box to use the algorithms defined on the **Default Connection** page ([Section A.1.2.1.3](#)), or define a MAC list using the arrow buttons. The MACs are tried in the order they are specified.

#### A.1.3.6 Defining Server Connections

On the **Server** tab, you can define advanced server connection settings for the profile.





**Figure A.16. Defining server connection settings for the profile**

### Use Defaults

Select the check box to use the values defined on the **Default Connection** page ([Section A.1.2.1.4](#)) for the server connection settings.

### Idle timeout

Specify how long idle time (after all connection channels are closed) is allowed for a connection before automatically closing the connection. The default is 5 seconds. Setting a longer time allows the connection to the server to remain open even after a session (for example, SSH Tectia terminal GUI) is closed. During this time, a new session to the server can be initiated without re-authentication. Setting the time to 0 (zero) terminates the connection immediately when the last channel to the server is closed.

### TCP connection timeout

Specify for how long a TCP connection will be attempted to a Secure Shell server. Define the timeout in seconds. After the defined time the TCP connection will be released in case the remote server is down or unreachable. Setting the value as 0 (zero) means that the default system TCP timeout will be used.

### Keepalive interval

Specify an interval (in seconds) for sending keepalive messages to a Secure Shell server. The default is 0, meaning that no keepalive messages are sent.

### Exclusive connection

Select this check box if you want that the profile always opens a new connection, instead of reusing a currently open connection.

## Show server banner

Select the check box if you want to have the server banner message file (if it exists) visible to users before login.

### A.1.3.7 Defining Proxy Settings

On the **Proxy** tab, you can select proxy settings for the profile.

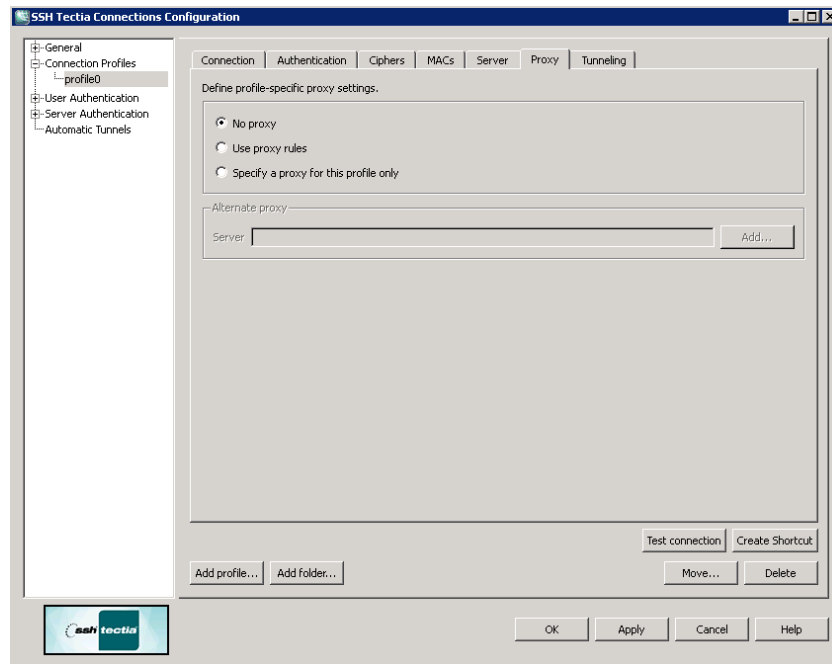


Figure A.17. Defining proxy settings for the profile

#### No proxy

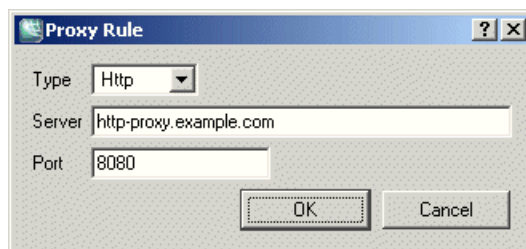
Select this option if you do not want to use a proxy.

#### Use proxy rules

Select this option to use the proxy rules defined in the **General** settings **Proxy Rules** page ([Section A.1.2.2](#)).

#### Specify a proxy for this profile only

Click **Add** to add a new proxy definition for this profile.



**Figure A.18. Defining alternate proxy for the profile**

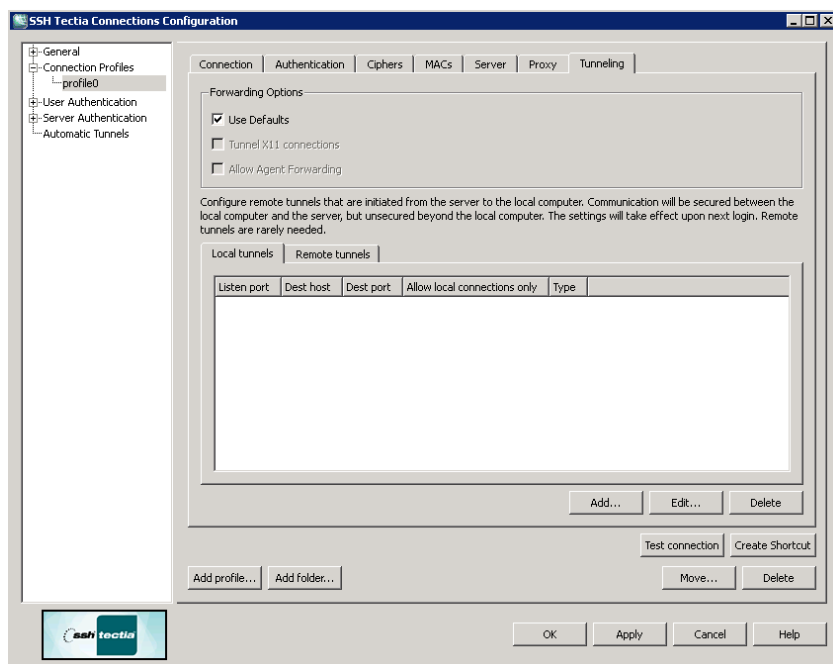
Select the **Type** of the rule. The type can be **Direct**, **Socks4**, **Socks5**, or **Http**.

For other types than direct, enter the address of the proxy **Server** and **Port**.

### A.1.3.8 Defining Tunneling

Tunneling, or port forwarding, is a way of forwarding otherwise unsecured TCP traffic through an encrypted Secure Shell connection (tunnel). You can secure for example POP3, SMTP, and HTTP connections that would otherwise be unsecured.

The tunneling settings for the connection profile are configured using the **Tunneling** tab. Any changed tunneling settings will take effect the next time you log in.



**Figure A.19. Defining tunneling through a profile**



## Note

The client-server applications using the tunnel will carry out their own authentication procedures (if any) the same way they would without the encrypted tunnel.

### A.1.3.8.1 Forwarding Options

It is possible to define separately for each connection profile whether X11 and/or agent forwarding are enabled, or whether the general default forwarding settings are applied to the profile.

#### Use Default

Select this option to make the profile follow the default settings for X11 and agent forwarding defined on the **Defaults - Tunneling** tab ([Section A.1.2.1.5](#)).

#### Tunnel X11 connections

Select the checkbox to allow this connection profile to use X11 forwarding.

SSH Tectia Client can securely tunnel (forward) X11 graphic connections from the remote host computer to an X Windows server running on the local computer.



## Note

A prerequisite for X11 tunneling is that you have an X emulator (such as eXceed or Reflection X) running in passive mode on the Windows computer.

To tunnel (forward) X11 traffic, do the following actions:

1. Install an X server (X emulation) program on Windows (eXceed, Reflection X, or the like).
2. Start SSH Tectia Client.
3. Select the **Tunneling** tab of the Connection Profiles page and make sure that the **Tunnel X11 connections** check box is selected.
4. Save your settings for SSH Tectia Client.
5. Restart SSH Tectia Client and log into the remote host.
6. Start the X server (X emulation) program.
7. To test the tunneling, run `xterm` or `xclock` from SSH Tectia Client.

For more information, see [Section 7.3](#).

#### Allow Agent Forwarding

Select the check box to allow this connection profile to use agent forwarding on the client side.

In agent forwarding, Secure Shell connections and public-key authentication data are forwarded from one server to another without the user having to authenticate separately for each server.

For more information, see [Section 7.4](#).

#### A.1.3.8.2 Local Tunnels

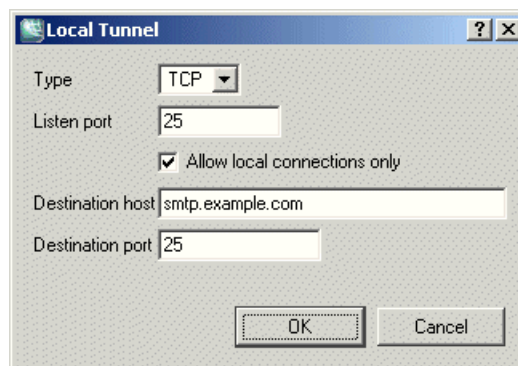
There are two types of tunnels that can be defined for application tunneling, local (outgoing) tunnels and remote (incoming) tunnels.

Local tunnels protect TCP connections that your local computer forwards from a specified local port to a specified port on the remote host computer you are connected to. It is also possible to forward the connection beyond the remote host computer, but the connection is encrypted only between SSH Tectia Client and SSH Tectia Server.

Remote tunnels protect TCP connections that a remote host forwards from a specified remote port to a specified port on your local computer.

To edit local tunnel definitions, click the **Local tunnels** tab.

To add a new local tunnel, click **Add**. The **Local Tunnel** dialog box opens.



**Figure A.20. Defining a local tunnel**

The following fields are used to define a local tunnel:

- **Type:** Select the type of the tunnel from the drop-down list. Valid choices are TCP and FTP. If you are tunneling an FTP connection, set the tunnel type as FTP. For other protocols, set the tunnel type as TCP.



#### Note

If the Secure Shell server and the FTP server are located on different computers, FTP tunneling works only if FTP is set to run in passive mode. If the Secure Shell server and the FTP server are located on the same computer, tunneling works regardless of whether FTP is running in passive or active mode.

- **Listen port:** This is the number of the local port which the tunnel listens to or captures.



### Note

The protocol or application that you wish to create the tunnel for may have a fixed port number (for example 143 for IMAP) that it needs to use to connect successfully. Other protocols or applications may require an offset (for example 5900 for VNC) that you will have to take into an account.

- **Allow local connections only:** Select this option if you want to allow only local connections to be made. This means that other computers will not be able to use the tunnel created by you. By default, only local connections are allowed. This is the right choice for most situations.

Consider the security implications carefully if you decide to also allow outside connections.

- **Destination host:** This field defines the destination host for the tunneling. The default value is `localhost`.



### Note

The destination host is resolved by the Secure Shell server, so here `localhost` refers to the Secure Shell server host you are connecting to.

- **Destination port:** The destination port defines the port that is used for the forwarded connection on the destination host.

To edit a tunnel definition, select a tunnel from the list and click **Edit**. The **Local Tunnel** dialog opens.

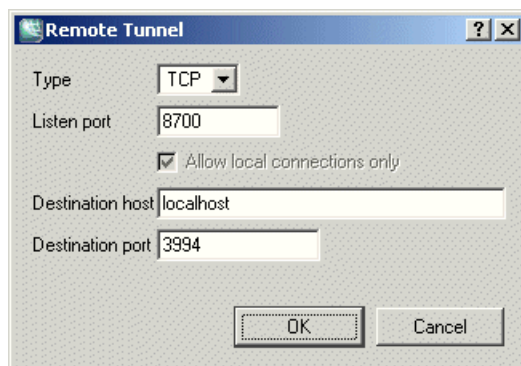
To delete a tunnel definition, select a tunnel from the list and click **Delete** to remove a tunnel. Note that the selected tunnel will be removed immediately, with no confirmation dialog.

For more information on local tunnels, see [Section 7.1](#).

#### A.1.3.8.3 Remote Tunnels

Remote (incoming) tunnels protect TCP connections that the remote host forwards from a specified remote port to the specified port on your local computer.

Click the **Remote tunnels** tab to edit incoming tunnel definitions. Click **Add** to open the **Remote Tunnel** dialog box.



**Figure A.21. Defining a remote tunnel**

The following fields are used to define a remote tunnel:

- **Type:** Select the type of the tunnel from the drop-down list. Valid choices are TCP and FTP. FTP tunneling is applicable with SSH Tectia ConnectSecure, only.
- **Listen port:** Enter the port that the tunnel listens to or captures from the remote host computer.



### Note

Privileged ports (below 1024) can be forwarded only when logging in with root privileges on the remote host computer.

- **Destination host:** Define the destination host for the port forwarding. The default value is localhost.



### Note

Here localhost refers to your local computer. Also note that if the connection from the remote host computer is forwarded beyond your local computer, that connection is unsecured.

- **Destination port:** Define the port that is used for the forwarded connection on the destination host.

To edit a tunnel definition, select a tunnel from the list and click **Edit**. The **Remote Tunnel** dialog opens.

To delete a tunnel definition, select a tunnel from the list and click **Delete** to remove a tunnel. Note that the selected tunnel will be removed immediately, with no confirmation dialog.

For more information on remote tunnels, see [Section 7.2](#).

## A.1.3.9 Defining Windows Settings

The type of the SSH Tectia window that is opened initially is configured using the **Windows** tab. The selected GUI version, **Terminal** or **File Transfer**, will be opened first when this profile is accessed.

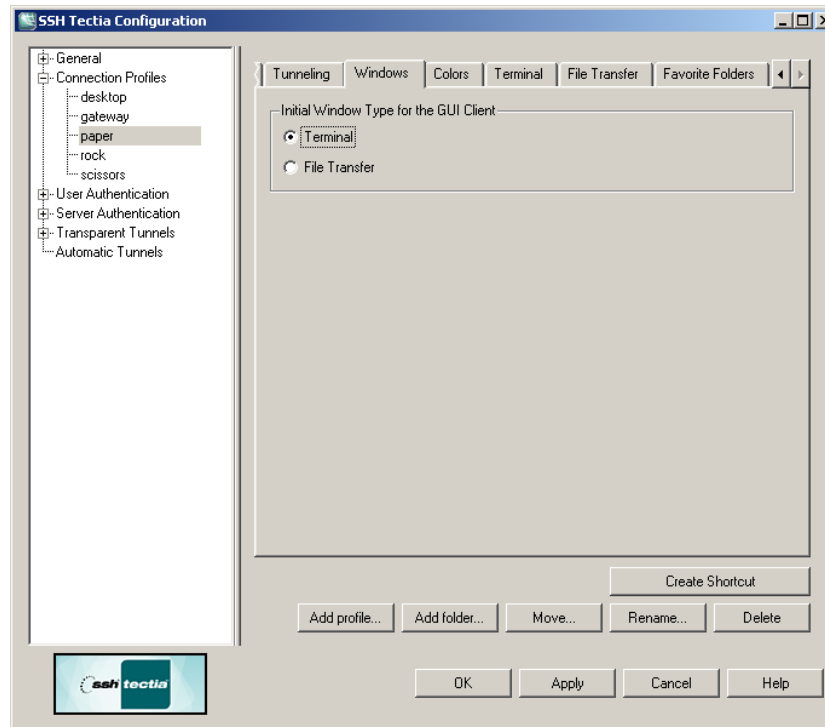


Figure A.22. Defining initial SSH Tectia window type



## Note

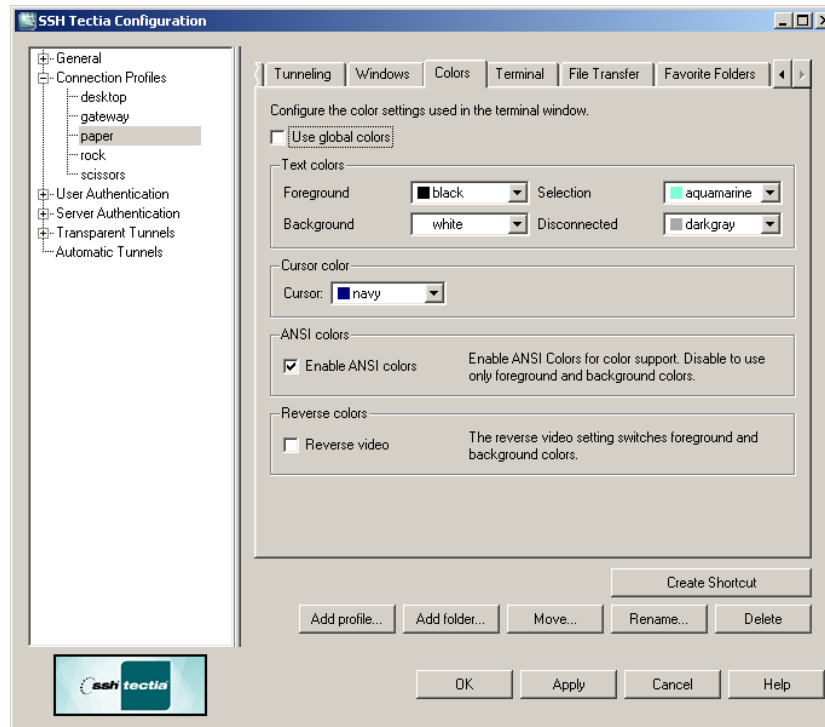
When a profile is added from the SSH Tectia GUI using the **Add Profile** option, the initial window type of the new profile is automatically set to be the same as in the current GUI view.

### A.1.3.10 Defining Color Settings

The colors used in the SSH Tectia Terminal GUI can be selected using the **Colors** page.

The color settings can be defined either globally or per profile. When colors are defined in SSH Tectia terminal Global Settings, the **Use Global Colors** option is not available, but the color settings will affect all connection profiles. See [Section B.1.3](#).





**Figure A.23. Defining SSH Tectia terminal colors**

**Use Global Colors:** Select this check box if you want to apply the global color settings to this profile. When this check box is selected, you cannot modify the color settings.

### Text Colors

The text colors affect the terminal window background color and the color of text in both a connected window and a disconnected window.

- **Foreground:** Select the desired foreground color from the drop-down menu. Foreground color is used for text in a window that has a connection to a remote host computer. You can select from sixteen colors. Black is the default foreground color.
- **Background:** Select the desired background color from the drop-down menu. You can select from sixteen colors. White is the default background color.
- **Selection:** Select the desired background color for mouse-selected texts from the drop-down menu. You can select from sixteen colors. Aquamarine is the default selection color.
- **Disconnected:** Select the desired foreground color for terminal windows that have no connection to a remote host computer. You can select from sixteen colors. Gray is the default foreground color for a disconnected terminal window.

## Cursor Color

Select the desired cursor color from the drop-down menu. You can select from sixteen colors. Navy is the default cursor color.

## ANSI Colors

With ANSI control codes it is possible to change the color of text in a terminal window. With the ANSI Colors setting you can select to use this feature. Even if you disable ANSI colors, you can still select your favorite text and background colors to be used in the terminal window.

Select the **Enable ANSI Colors** check box to allow ANSI colors to be used in the terminal window. By default, ANSI colors are selected.

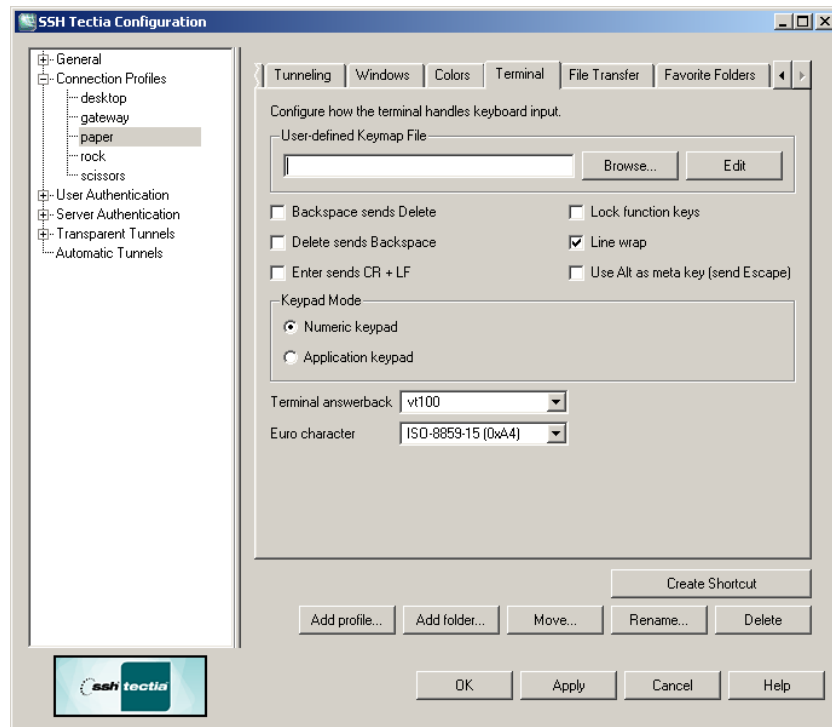
## Reverse Colors

By reversing the display colors you can quickly change the display from positive (dark on light) to negative (light on dark) to improve visibility.

Select the **Reverse Video** check box to change the foreground color into background color and vice versa. This setting affects the whole terminal window when you click **OK**.

### A.1.3.11 Defining Terminal Settings

The settings used for the SSH Tectia Client terminal are configured using the **Terminal** tab. Keyboard mappings take effect when you start a new connection or reset the terminal.



**Figure A.24. Defining SSH Tectia terminal settings**

## User Defined Keymap File

Use this option to create additional keyboard shortcuts or to modify the existing ones. The additional key mappings are saved into a separate file with the `.sshmap` file extension. The current keymap file is displayed in the text field.

You can modify the current key mappings by clicking **Edit** to open the Keymap Editor dialog.

If you have defined an alternative keymap settings file, you can load it by typing the path and file name in the text field, or by clicking on the button on the right-hand side of the text field. Clicking the button will open an Open dialog that allows you to locate an alternative keymap file.

Select the **Backspace sends Delete** check box if you want to map the Backspace key to the Delete operation.

Select the **Delete Sends Backspace** check box if you want to map the Delete key to the Backspace operation.

Select the **Enter sends CR + LF** check box if you want to map the Enter key to send the carriage return (CR) and line feed (LF) characters. Otherwise only the line feed character will be sent.

Select the **Lock Function Keys** check box if you want to lock the function keys.

Select the **Line Wrap** check box if you want the text lines to wrap at the terminal window edge. By default, line wrapping is on.

Select the **Use Alt as meta key (send Escape)** check box if you want the Alt key to function as the meta key in the same way as the Escape key. If this option is selected, you can for example press the Alt+X key combination to simulate the Escape followed by X.

## Keypad Mode

Select how you want the numeric keypad on the right-hand side of the regular keyboard to function.

Select **Numeric Keypad** to use the keypad to enter numbers.

Select **Application Keypad** to use the keypad for application control (with the keypad keys functioning as cursor keys, Home, End, Page Up, Page Down, Insert and Delete).

## Terminal answerback

Use the **Terminal answerback** drop-down list to select the same terminal answerback mode that is used by the SSH Tectia Server related to the profile.

## Euro character

Use the **Euro character** drop-down list to select the support mode for the euro character (€).

The supported options are Windows (where euro is mapped as 0x80) and ISO 8859-15 (euro mapped as 0xA4). Select the same character set that is used by the SSH Tectia Server related to the profile.

Note however that enabling the euro character support will disable the 8-bit terminal control codes.

### A.1.3.12 Defining File Transfer Settings

The **File Transfer** tab defines which files are transferred using ASCII mode and which newline conventions are applied.

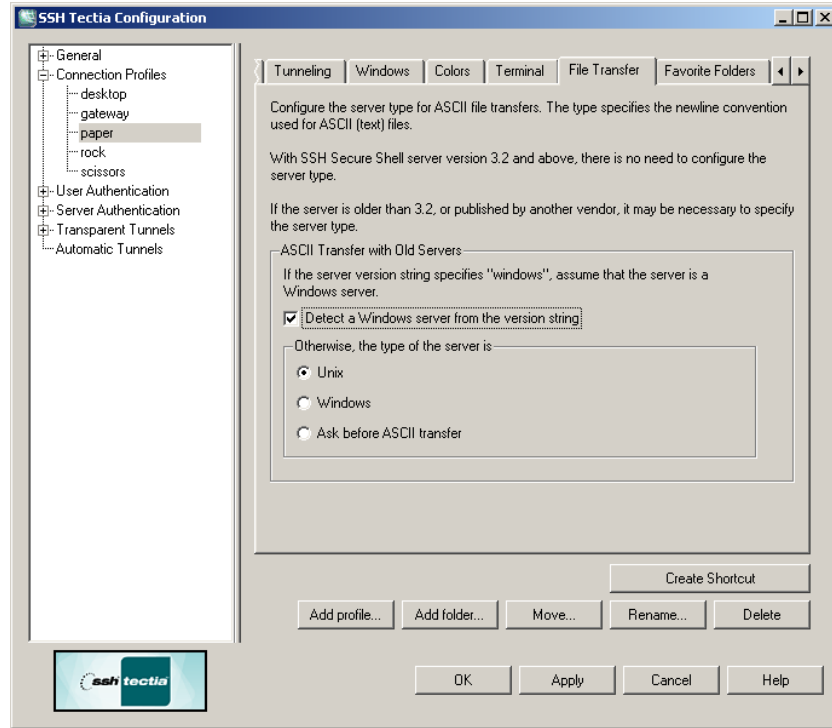


Figure A.25. Defining SSH Tectia file transfer settings

#### ASCII transfer with old servers

**Detect Windows server from the version string:** Secure Shell client and server exchange version strings when setting up the connection. Select this check box to automatically detect Windows servers and use the correct setting for them. For this feature to work correctly, the Windows server has to specify "windows" in its version string.

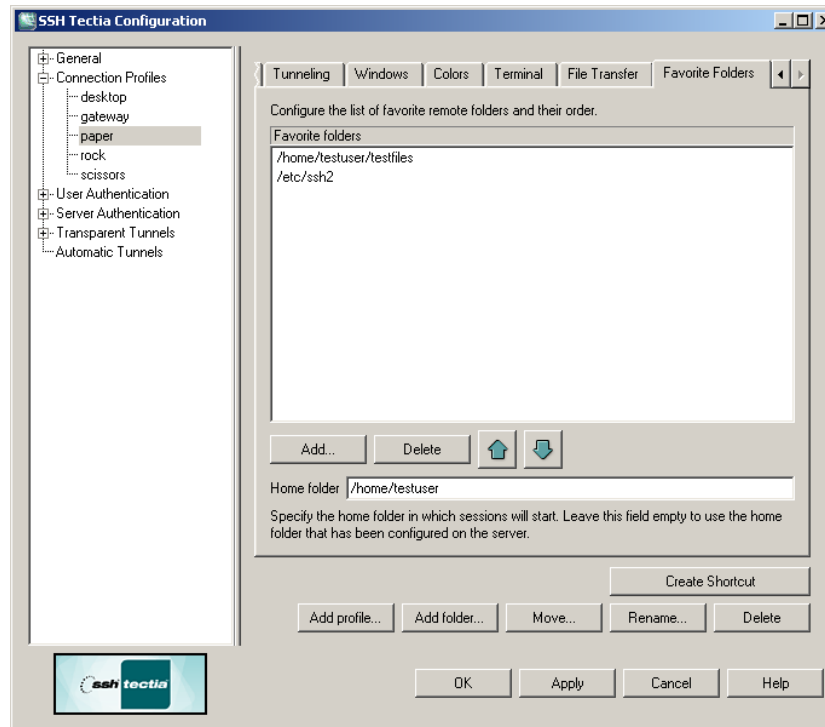
Select the **Unix** check box to use Unix compatible line breaks (LF).

Select the **Windows** check box to use Windows compatible line breaks (CRLF).

Select the **Ask before ASCII transfer** check box to make SSH Tectia Client ask you to specify the server type before each ASCII file transfer.

### A.1.3.13 Defining Favorite Folders

In the **Favorites Folders** tab, you can create a list of commonly used remote directories. These favorites can then be easily selected from a drop-down menu in the file transfer window.



**Figure A.26. Defining favorite remote folders for file transfer**

### Favorite Folders

This list contains the favorite folders you have defined for the current connection profile. You can add, remove, and sort the favorites by using **Add**, **Delete**, and the arrow buttons below the list.

If you are defining a remote favorite that is located on a Windows Secure Shell server, the folder on the Windows server must be specified as follows: `/drive/folder/subfolder`.

A valid favorite folder definition would be, for example:

```
/C/Documents and Settings/All Users/Desktop
```

### Home Folder

In the **Home Folder** field you can enter the directory where any new SFTP connections associated with this profile will start. If you leave the field empty, new connections will use the remote home folder that has been specified for your user account on the remote host computer.

## A.1.4 Defining User Authentication

Under **User Authentication**, you can configure settings related to public-key and certificate authentication. See [Section A.1.4.1](#) and [Section A.1.4.2](#).

To enable or disable public-key authentication, see [Section A.1.2.1](#) and [Section A.1.3.2](#).

### A.1.4.1 Managing Keys and Certificates

On the **Keys and Certificates** page, you can add key and certificate files used in user authentication, generate a new key, upload a key to a server, or change the passphrase for a key.

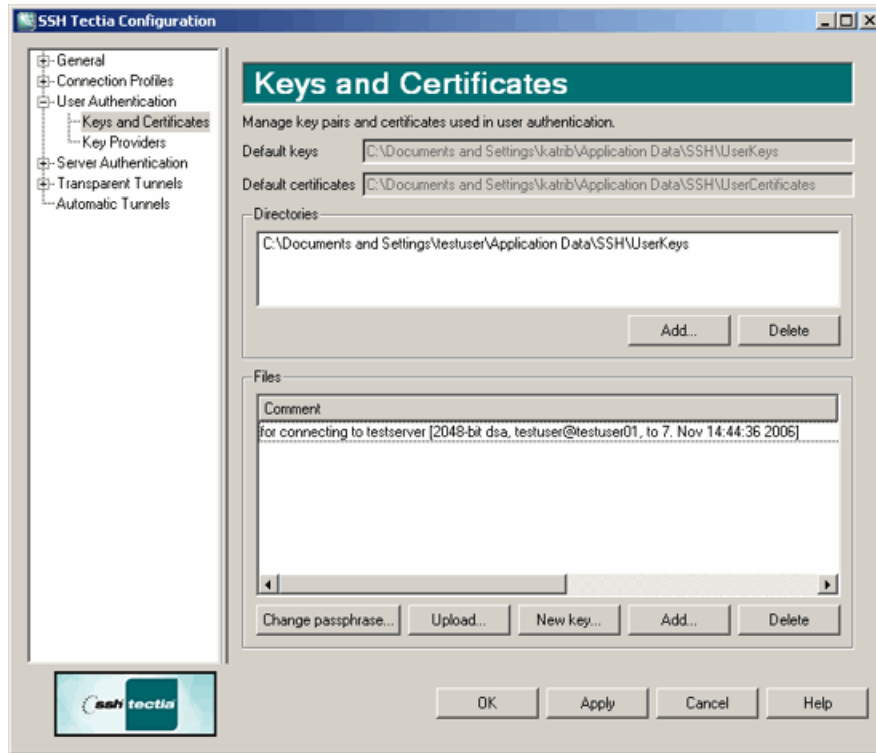


Figure A.27. Defining keys and certificates

#### Default keys

The default location of user keys.

#### Default certificates

The default location of user certificates.

#### Directories

Use the **Add** button to add a directory of keys, **Delete** to remove.

#### Files

All public keys known to SSH Tectia Client are listed in this field. You can modify the key details by selecting a key file in the list and clicking a button at the bottom.

Click **Change passphrase** to change the passphrase of a selected key.

Click **Upload** to upload the key to a remote server. See also [Section 5.4.3.2](#).

Click **New key** to start the key generation wizard. For a description of the wizard, see [Section A.1.3.3](#).

Click **Add** to import an existing key or certificates from elsewhere in your system.

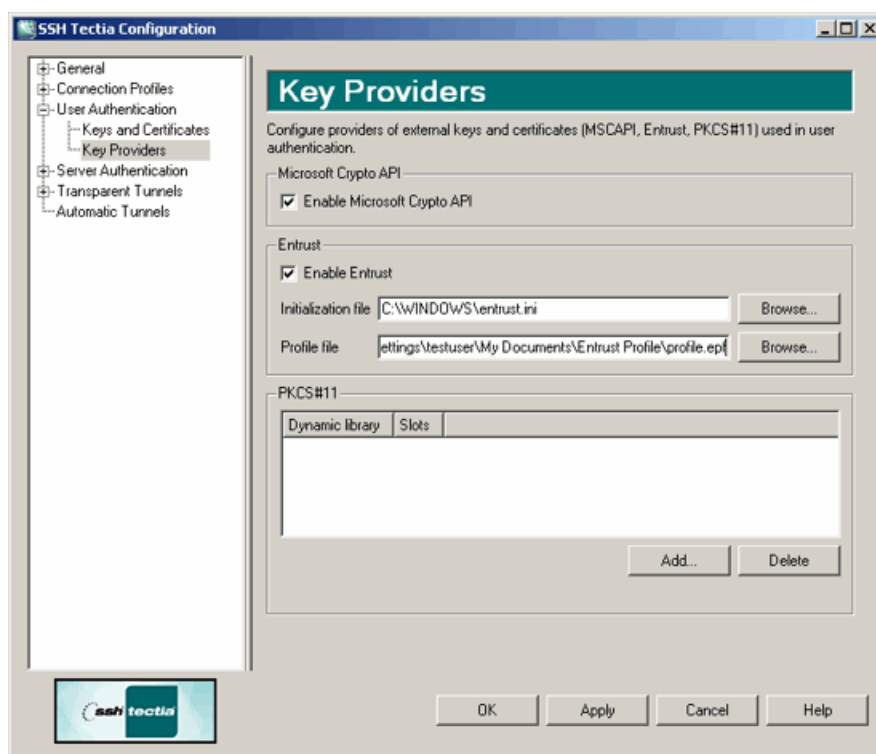
Click **Delete** to remove a selected key.

## **i** Note

The user-specific `Application Data` directory, where the public key files are stored, is hidden by default. To view hidden directories, change the setting in Windows Explorer. For example, on Windows XP, select **Tools** → **Folder Options** on the menu, click the **View** tab, and select **Show hidden files and folders**.

### A.1.4.2 Managing Key Providers

On the **Key Providers** page you can define the settings of external key providers used in user authentication. Available key providers are MSCAPI, Entrust, and PKCS#11.



**Figure A.28. Defining key providers**

#### Microsoft Crypto API

SSH Tectia Client can access keys via Microsoft Crypto API (MSCAPI). MSCAPI is a standard cryptographic interface used in Microsoft Windows systems.

Microsoft Crypto API (MSCAPI) providers can be enabled by selecting the **Enable Microsoft Crypto API** check box. If you enable the MSCAPI providers, you can use software keys and certificates created by Microsoft applications.

### Entrust

Select the **Enable Entrust** check box to enable using Entrust. Entrust is available on Microsoft Windows.

Enter the **Initialization file** (\*.ini) and **Profile file** (\*.epf).

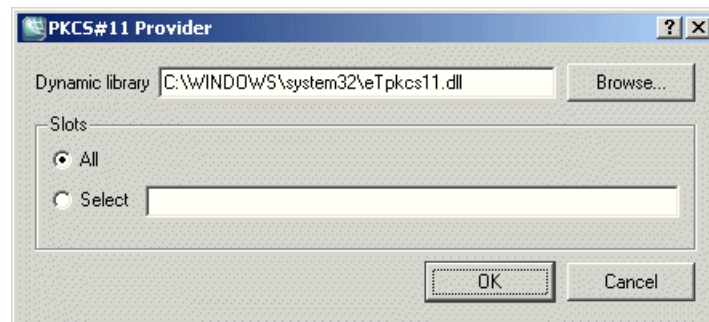
By using the Entrust provider, SSH Tectia Client can utilize keys and certificates stored in an Entrust profile file (.epf). The initialization file includes the basic Entrust PKI configuration (for example the CA address).

When the provider is enabled for the first time, Entrust Entelligence will prompt for your Entrust password. As long as the Entrust provider is enabled, the password is asked each time SSH Tectia Client is started.

### PKCS#11

By using the PKCS#11 provider, SSH Tectia Client can use keys and certificates stored in PKCS#11 tokens (for example, smart cards or USB tokens).

Click **Add** to define a PKCS#11 provider.



**Figure A.29. Defining a PKCS#11 provider, Aladdin eToken DLL path shown as an example**

Use the **Dynamic library** to define a dynamic library containing the PKCS#11 driver.

Use the **Slots** to define slots. A slot is a logical reader that potentially contains a token. Slots are manufacturer-specific. They are defined with an integer. Examples: "0,1", "0-3, !2", "2".

## A.1.5 Defining Server Authentication

Under **Server Authentication**, you can define how SSH Tectia Client authenticates remote server hosts.

- To use public keys in server authentication, define the settings as described in [Section A.1.5.1](#).
- To apply certificates, define the settings as described in [Section A.1.5.2](#).



- Settings required for LDAP usage are described in [Section A.1.5.3](#).
- To define regular intervals for fetching certificate revocation lists (CRLs), see [Section A.1.5.4](#).

### A.1.5.1 Managing Host Keys

On the **Host Keys** page, you can add new public host keys, define the host key acceptance policy, and view and manage known host keys used in server authentication. Known host keys mean keys already stored to the user-specific %APPDATA%\SSH\Hostkeys directory.

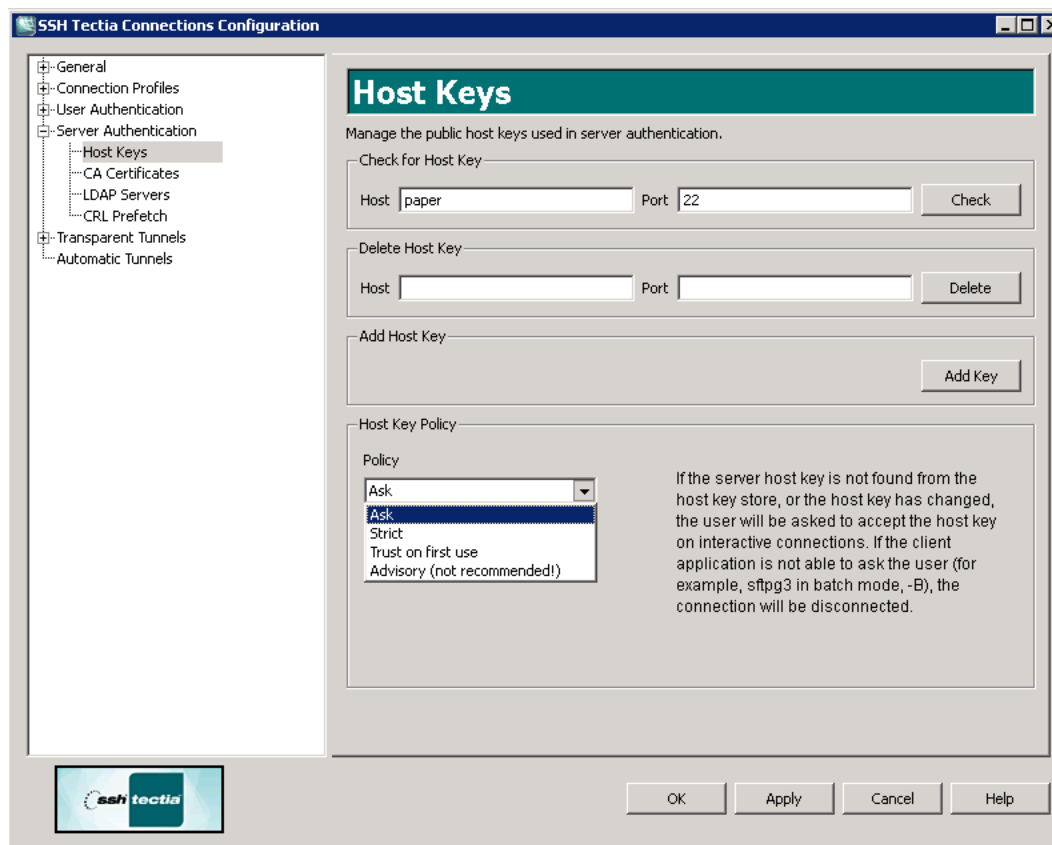


Figure A.30. Defining server host keys settings



#### Note

The host key policy settings have changed in version 6.1.4. SSH Tectia Connections Configuration GUI updates the user-specific configuration automatically to use the new policy based on the old **Strict host key checking**, **Accept unknown host keys**, and **Always show host key prompt** settings. The interpretation of the old policy to the new policy is shown in [Table A.2](#).

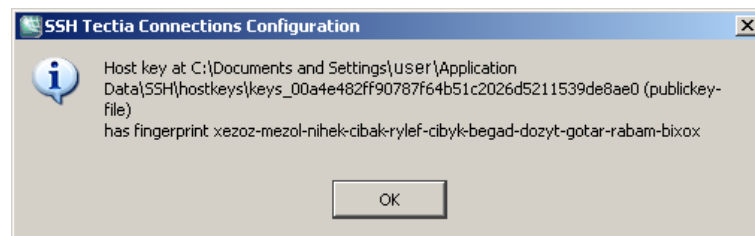
The **Host Keys** view includes the following options:

### Check for Host Key

You can check if a public host key of a remote server exists on your client, and view its fingerprint. To check the host key, enter the name of the server in the **Host** field and the listener port number in the **Port** field, and click **Check**.

Note that wildcard characters are not allowed, specify the exact host name and port.

When a public host key for the specified server is found on the client, a dialog-box shows where the host key is stored and what is the fingerprint of the public key. The fingerprint is shown in the SSH Babble format, consisting of a series of pronounceable five-letter words in lower case and separated by dashes. See an example below.



**Figure A.31. Server public host key information**

For more information on server public host keys, see [Section 5.1](#).

### Delete Host Key

In case you want to delete a known public host key from the client side, enter the name of the relevant server in the **Host** field and the listener port number in the **Port** field, and click **Delete**.

A dialog box appears asking you to confirm or to cancel the deleting of the host key.

### Add Host Key

Click the **Add Key** button to add a new host key to your known host keys directory. The Connection Broker opens a file manager view where you can browse to the key location and select the host key you want to copy.

### Host Key Policy

Select the policy you want to apply to the checking of server host keys and to the handling of unknown server host keys.



#### Note

This setting strongly affects the security of the client side host.

The options are:

- **Ask - the default** - the user will be asked to verify and accept the server public host keys, if the keys are not found in the host key store or if the keys have changed. The user can decide whether the key

is to be stored to %APPDATA%\SSH\Hostkeys, or used once without storing it, or cancelled. Connection is allowed only to a server whose host key is either found in the known host keys directory or accepted by the user currently.

This policy requires an interactive connection to get a response from the user. If the **Ask** option is applied on a non-interactive connection, the connection will be closed.

- **Strict** - the connection to the server will be allowed only if the host key is found in the user's known host keys storage. Otherwise, the connection will be closed. This option expects that all acceptable server host keys have already been stored on the client. No new host key's will be stored, and connections to any servers that have changed host keys will be closed.

This option can be used on non-interactive connections, once the host keys have been received by other means. This policy provides maximum protection against man-in-the-middle attacks.

- **Trust on first use** - new host keys are stored without prompting the user to accept them. Connections to servers offering a changed host key will be closed. This policy should be used only when server host keys cannot be added to the key storage by any other means.
- **Advisory** - *not recommended* - new host keys are stored without prompting the user to accept them, and connections are allowed also to servers offering a changed host key. Changed keys are not stored on the client, and data about opening connections with them are logged, provided that logging is enabled on the Connection Broker.

If you choose this policy, make sure the Connection Broker has logging activated in the **General - Logging** view, see [Section A.1.2.3](#). Then you have the possibility to detect any connections with changed host keys in the logs.



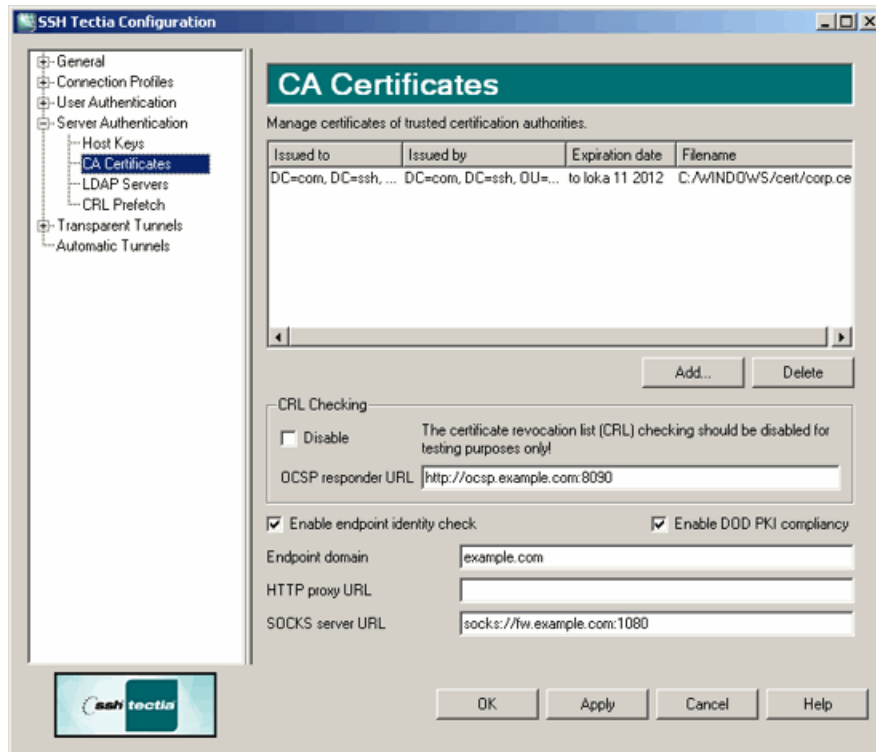
### Caution

Consider carefully before you activate the **Advisory** policy, as it practically disables server authentication and makes the connection vulnerable to active attackers.

## A.1.5.2 Managing CA Certificates

On the **Certificates** page, you can manage trusted CA certificates.

For more information on server certificate authentication, see [Section 5.2](#).



**Figure A.32. Defining CA certificates**

The following fields are displayed on the CA certificate list:

- **Issued to:** The certification authority to whom the certificate has been issued.
- **Issued by:** The entity who has issued the CA certificate.
- **Expiration date:** The date that the CA certificate will expire.
- **Filename:** The file containing the CA certificate.

#### CRL Checking

Select the **Disable** check box to prevent the use of a certificate revocation list (CRL). A CRL is used to check if any of the used server certificates have been revoked.

#### Note

Disabling CRL checking is a security risk and should be done for testing purposes only.

#### OCSP responder URL

The OCSP Responder Service provides client applications a point of control for retrieving real-time information on the validity status of certificates using the Online Certificate Status Protocol (OCSP).

For the OSCP validation to succeed, both the end-entity (=Secure Shell server) certificate and the OSCP responder certificate must be issued by the same CA. If the certificate has an `Authority Info Access` extension with an OSCP Responder URL, it is only used if there are no configured OSCP responders. It is not used if any OSCP responders have been configured.

If an OSCP responder is defined in the configuration file or in the certificate, it is tried first; only if it fails, traditional CRL checking is tried, and if that fails, the certificate validation returns a failure.

### Enable endpoint identity check

Specifies whether the client will verify the server's hostname or IP address against the **Subject Name** or **Subject Alternative Name** (DNS Address) specified in the server host certificate. By default, **Enable endpoint identity check** is enabled.

If this check box is not selected, the fields in the server host certificate are not verified and the certificate is accepted based on the validity period and CRL check only.



### Caution

Disabling the endpoint identity check on the client is a security risk. Then anyone with a certificate issued by the same trusted CA that issues the server host certificates can perform a man-in-the-middle attack on the server.

### Enable DOD PKI compliancy

This element defines whether the certificates are required to be compliant with the DoD PKI (US Department of Defense Public-Key Infrastructure).

### Endpoint domain

Specify the default domain used in the end-point identity check. This is the default domain part of the remote system name and it is used if only the base part of the system name is available.

If the default domain is not specified, the end-point identity check fails, for example, when a user tries to connect to a host "tower" giving only the short hostname and the certificate contains the full DNS address "tower.example.com".

### HTTP proxy URL

Specify the HTTP proxy used when making LDAP or OSCP queries for certificate validity.

The format of the address is "http://username@proxy\_server:port/network/netmask,network/netmask... ". The `network/netmask` part is optional and defines the network(s) that are connected directly (without the proxy).

### SOCKS server URL

Specify the SOCKS server used when making LDAP or OSCP queries for certificate validity.

The format of the address is "socks://username@socks\_server:port/network/netmask, network/netmask... ". The network/netmask part is optional and defines the network(s) that are connected directly (without the SOCKS server).

### A.1.5.3 Managing LDAP Server Settings

On the **LDAP Servers** page, you can define LDAP servers used for fetching CRLs and/or subordinate CA certificates based on the issuer name of the certificate being validated.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be verified if the point exists.

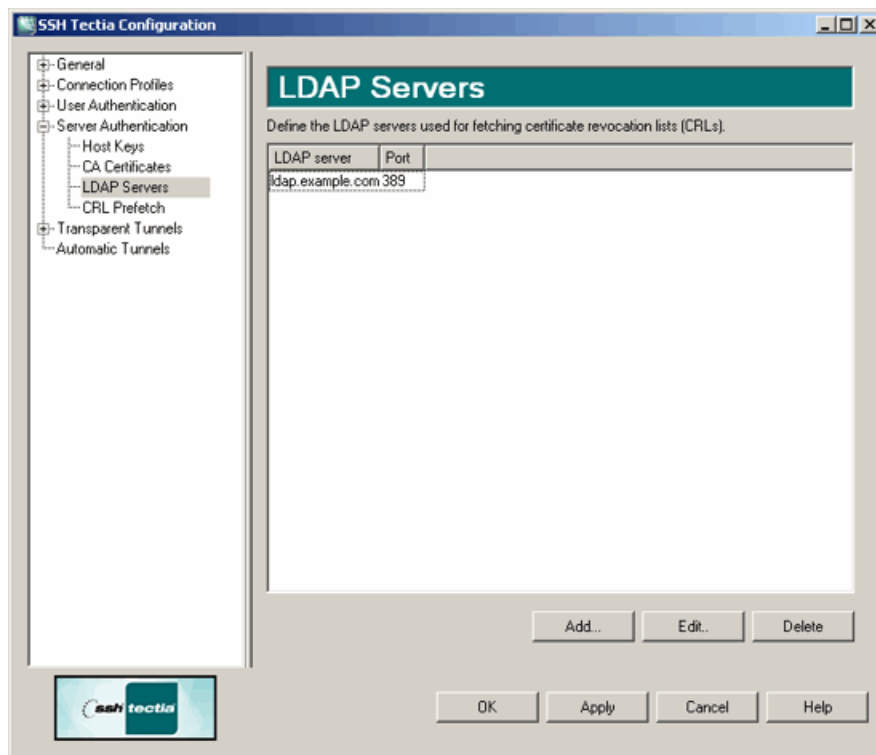


Figure A.33. Defining LDAP servers

To add an LDAP server, click the **Add** button. Define the hostname and port for the server.



Figure A.34. Adding an LDAP server

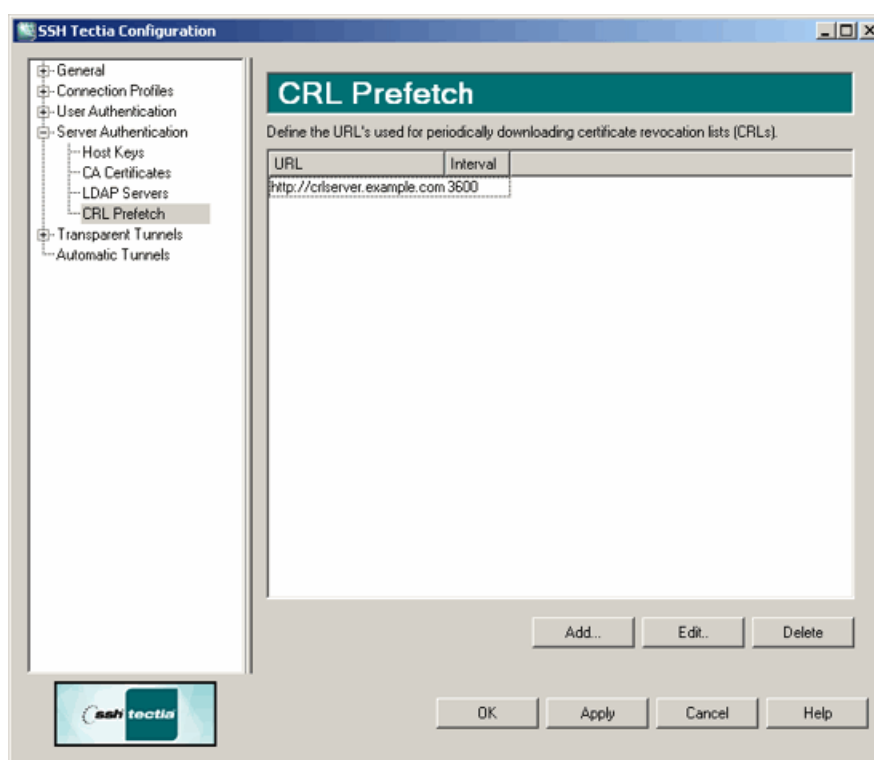
To edit an LDAP server, select the server from the list and click **Edit**.

To delete an LDAP server, select the server from the list and click **Delete**.

### A.1.5.4 Managing CRL Prefetch Settings

On the **CRL Prefetch** page, you can define certificate revocation lists (CRLs) to be fetched from the defined location at regular intervals. The CRL distribution point can be either a standard format LDAP or HTTP URL, or it can refer to a file. The file format must be either binary DER or base64, PEM is not supported.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be verified if the point exists.



**Figure A.35. Defining CRL prefetch settings**

To add a CRL prefetch address, click **Add**. The **CRL Prefetch** dialog box opens.



**Figure A.36. Adding a CRL prefetch setting**

Enter the **URL** of the CRL distribution point and the **Interval** how often the CRL is downloaded and click **OK**. The default download interval is 3600 (seconds).

In case the CRL distribution point refers to a file, enter the file URL in this format:

```
file:///absolute/path/name
```

To edit an existing CRL prefetch setting, select the setting from the list and click **Edit**.

To delete an existing CRL prefetch setting, select the setting from the list and click **Delete**.

## A.1.6 Defining Transparent Tunnels

Under **Transparent Tunnels**, you can define the settings for transparent tunneling of applications using TCP or FTP services. For generic connection capture settings, see [Section A.1.6.1](#), and for defining the filter rules, see [Section A.1.6.2](#).

All settings are made in the Connection Broker configuration, so no modifications are required on the tunneled applications.

### A.1.6.1 Defining the Connection Capture Settings

On the **Connection Capture** page, you can define how transparent TCP tunneling captures the connections made by TCP-based applications.



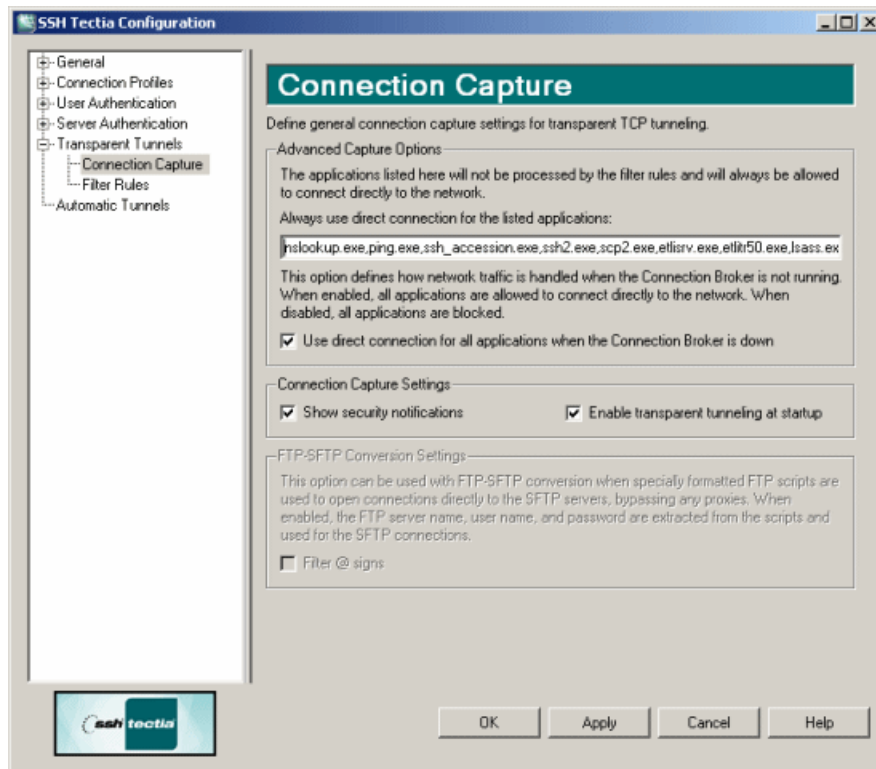


Figure A.37. Defining the transparent TCP tunneling settings

### Advanced Capture Options

Define the exceptional applications that will be allowed to use direct connection to the network instead of being captured and tunneled securely. These applications will not be processed by the filter rules.

The application names are handled case-insensitively. Make sure the process names include also the file extensions. You can check the correct name format in *Windows Task Manager*. Use commas but no spaces to separate the entries, for example:

```
ssh-client-g3.exe,nslookup.exe,ping.exe
```

The direct connection settings are not stored in the `ssh-broker-config.xml` file but directly in the Windows Registry, under `HKEY_LOCAL_MACHINE\SOFTWARE\SSH Communications Security\SSH Tectia Connector`

**Use direct connection for all applications when the Connection Broker is down:** Select this option if it is necessary to temporarily deactivate connection capturing so that it does not block network communications. If this option is un-selected, all applications will be blocked when the Connection Broker is down. When this option is selected (the default), all applications will be able to connect to the network when the Connection Broker is down. If users should only access the network using secure communications, leave this option disabled.

## Connection Capture Settings

**Show security notification:** Select this option to have a notification briefly displayed when a new application is secured with a FTP or TCP tunnel, and when the tunneling ends. The notification specifies the secured application, the destination, as well as the Secure Shell server used as the tunneling end point. A list of currently tunneled applications is shown in the SSH Tectia Status window (started via the short cut menu).



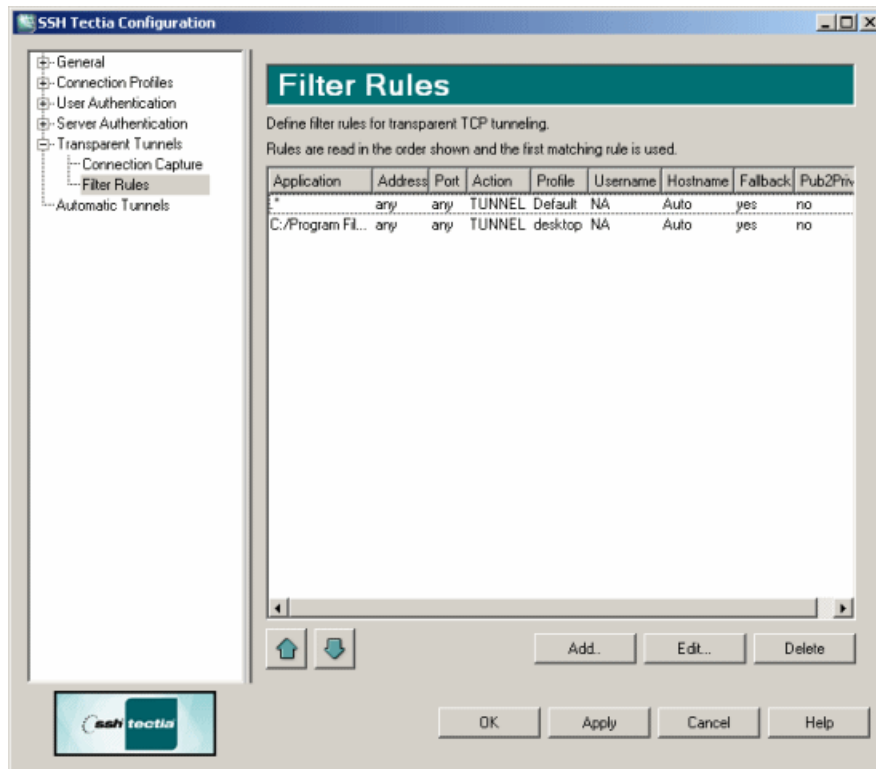
Figure A.38. Security notification

**Enable transparent tunneling at startup:** Select this option to activate the transparent TCP tunneling feature when Connection Broker starts up. To disable transparent TCP tunneling in future sessions, clear the **Enable transparent tunneling at startup** check box. Connection Broker reads this setting in the configuration when it starts up.

When this setting is selected, the text `Transparent tunneling enabled` will be shown in the SSH Tectia tray menu. The shortcut menu shows the current status of transparent TCP tunneling, and the feature can be temporarily disabled by unselecting `Transparent tunneling enabled` in the menu. The setting in the SSH Tectia tray menu is not saved in the configuration.

### A.1.6.2 Defining Filter Rules

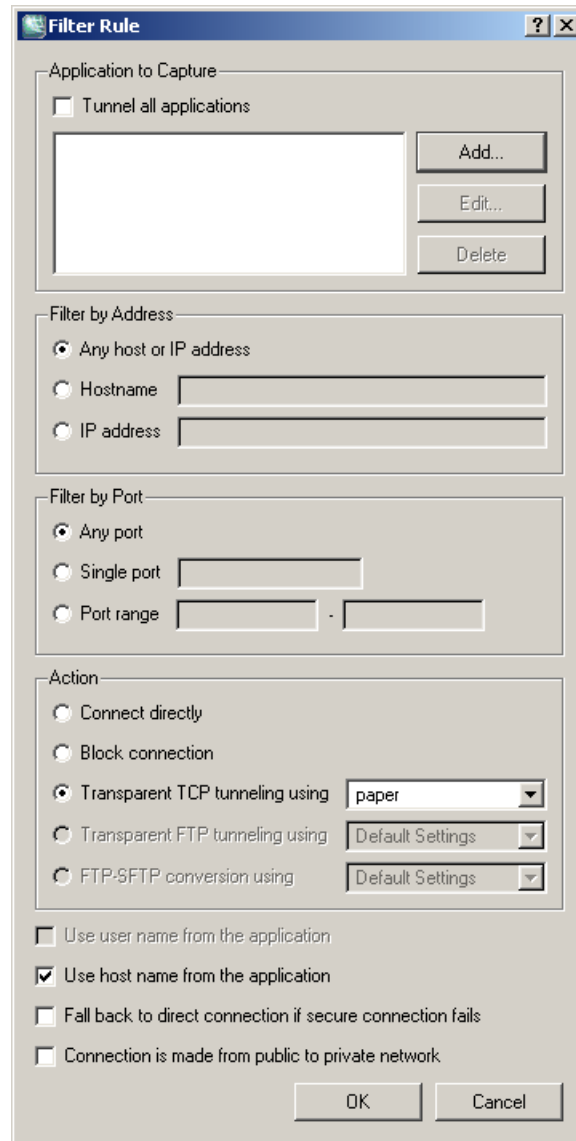
On the **Filter Rules** page, you can define the filters based on the characteristics of tunneled applications. The filters are used to select how and to which applications the transparent tunneling will be applied.



**Figure A.39. Defining filter rule settings**

When an application connects to a host, the filter rules are used to determine the correct action to apply to the connection. The filter list is scanned through searching for a filter that matches the connection. The first filter that matches the DNS or IP address of the connection is used. Filters are evaluated from top down. You can use the arrow buttons to organize the list.

Click the **Add** button to define a new filter rule in the **Filter Rule** dialog box. Click **Edit** to modify and **Delete** to remove existing filter rules.



**Figure A.40. Adding a new filter rule**

### Application to Capture

**Tunnel all applications:** Select this option to capture all connections initiated by TCP-based applications.

To specify only some applications to be captured, click **Add** and enter the name of an application or locate the application with **Browse...**. You can list several applications. The path and application name must be given using regular expressions following the egrep syntax. If you use the **Browse**, the GUI enters the applications automatically in the correct format. For information on the syntax, see [Appendix D](#).

To modify or delete the listed applications, select the relevant application and click **Edit** or **Delete**.

### Filter by Address

Define hosts whose connections will be captured.

**Any host or IP address:** Select this option to capture the connections to all hosts.

**Host name:** Select this option to capture only the connections to individual hosts. Define the DNS address(es) of the host(s) in a comma-separated list. The SSH Tectia Client will resolve the IP address using a DNS query. The value can also be a regular expression.

**IP address:** Select this option to capture only the connections to the defined IP address(es). The value can also be a regular expression.

### Filter by Port

Define the ports whose connections will be captured.

**Any port:** Select this option to capture the connections of all ports.

**Single port:** Select this option to define only individual port(s) to be captured. Enter the port number(s) in a comma-separated list.

**Port range:** Select this option to define a range of port numbers whose connections will be captured.

### Action

**Connect directly:** Select this option to make the connection directly to the host without tunneling, using the host's IP address if it can be resolved. If it cannot be resolved, the connection fails.

**Block connection:** Select this option to block the connection. Applications usually inform the user that the connection is refused.

**Transparent TCP tunneling using:** Activates transparent TCP tunneling for the defined connections. Select from the drop-down menu whether the transparent TCP tunneling is used with the default settings, or through a connection profile. By default, the transparent TCP tunneling uses the destination host name received from the application that initiated the connection. When a profile is used, you can choose to use the destination host name and the user name defined in the profile, or those received from the application.

If the connection is made using a DNS name, the tunnel is created with the DNS name. This means that the actual DNS name resolution is done at the remote end, which enables tunneling connections to hosts that are not visible to the local machine. If the used port does not match a port or port range, the connection is direct.

**Transparent FTP tunneling using:** Activates transparent FTP tunneling for the defined connections. This is available with SSH Tectia ConnectSecure only.

**FTP-SFTP conversion using:** Activates FTP-SFTP conversion for the defined connections. This is available with SSH Tectia ConnectSecure only.

### Additional

**Use user name from the application:** This is not available with TCP tunneling.

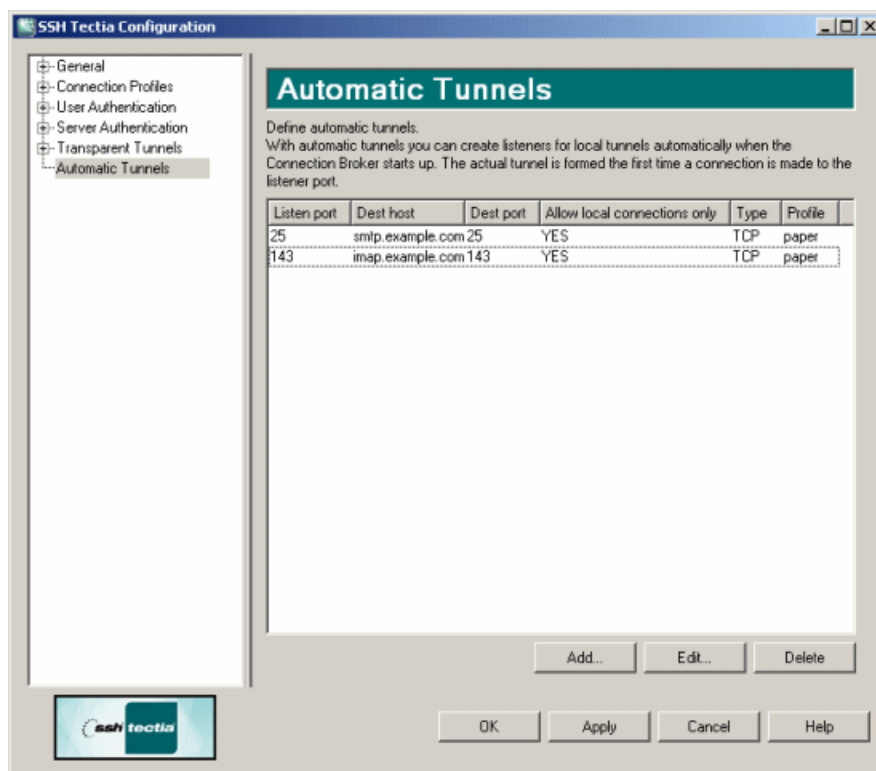
**Use host name from the application:** Select this option to make SSH Tectia Client resolve and use the host name sent by the application (instead of doing a DNS query) to establish a tunnel to the destination host. When the check box is not selected, a normal DNS query is made. By default, this setting is on for transparent TCP tunneling. When transparent TCP tunneling is made through a connection profile, you can choose to disable this setting.

**Fall back to direct connection if secure connection fails:** Select this option to allow a direct (unsecured plain-text) connection in case creating a tunnel fails or the connection to the Secure Shell server fails. If this is not selected, the Connection Broker will normally return a "host not reachable" error.

**Connection is made from public to private network:** Use this option if the connection is made from public network to a private network with its own address space. This setting specifies whether a pseudo IP address will be used when an IP address cannot be resolved by the Connection Broker. When the checkbox is not selected, a normal DNS query is made for the target hostname. When the checkbox is selected, the Connection Broker assigns a pseudo IP address for the target host and Secure Shell server will resolve the real IP address. This is needed because the name resolution for machines located in an internal network is not available from outside.

## A.1.7 Defining Automatic Tunnels

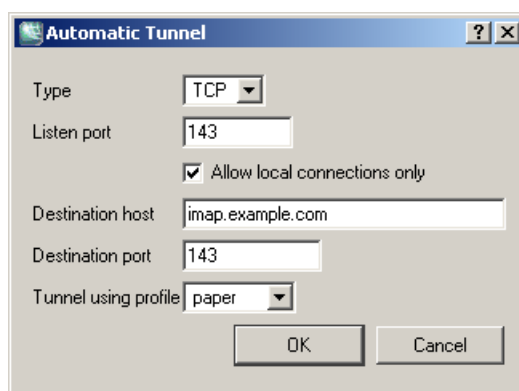
On the **Automatic Tunnels** page, you can create listeners for local tunnels that are started automatically when the Connection Broker starts up. The actual tunnel is formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.



**Figure A.41. Defining automatic tunnels**

When the Connection Broker starts, the list of the automatic tunnels is read, and the connection initiating applications will be matched to the rules defined here. The first setting that matches the connection will be used. The rules are evaluated from top down, and you can use the arrow buttons to organize the list.

Select **Automatic Tunnels** in the tree menu and click **Add** to open the **Automatic Tunnel** dialog box.



**Figure A.42. Adding a new automatic tunnel**

- **Type:** Select the type of the tunnel from the drop-down list. Valid choices are TCP and FTP.

- **Listen port:** This is the number of the local port that the tunnel listens to, or captures. Do not use a reserved port number.



### Note

The protocol or application that you wish to create the tunnel for may have a fixed port number (for example 143 for IMAP) that it needs to use to connect successfully. Other protocols or applications may require an offset (for example 5900 for VNC) that you will have to take into an account.

- **Allow local connections only:** Leave a check mark in this box if you want to allow only local connections to be made. This means that other computers will not be able to use the tunnel created by you. By default, only local connections are allowed. This is the right choice for most situations. You should carefully consider the security implications if you decide to also allow outside connections.
- **Destination host:** This field defines the destination host for the port forwarding. The default value is localhost.



### Note

The value of localhost is resolved by the Secure Shell server, so here `localhost` refers to the Secure Shell host you are connecting to.

- **Destination port:** The destination port defines the port that is used for the forwarded connection on the destination host.
- **Tunnel using profile:** Select the server to use for the tunnel.

To edit a automatic tunnel, select a tunnel from the list and click **Edit**.

To delete a automatic tunnel, select a tunnel from the list and click **Delete**.

For more information on tunneling, see [Section 7.1](#).

## A.2 Configuration File for Connection Broker

The elements of the XML-based Connection Broker configuration file `ssh-broker-config.xml` are described in [ssh-broker-config\(5\)](#).

### ssh-broker-config

ssh-broker-config -- SSH Connection Broker configuration file format



The Connection Broker configuration file `ssh-broker-config.xml` is used by SSH Tectia Client, Connect-Secure, and SSH Tectia MFT Events on Unix and Windows, and additionally by the SSH Tectia client tools on z/OS and z/Linux. The Connection Broker configuration file must be a valid XML file that follows the `ssh-broker-ng-config-1.dtd` document type definition.

## Connection Broker Files

The Connection Broker reads three configuration files (if all are available):

1. The `ssh-broker-config-default.xml` file is read first. It holds the factory default settings. It is not recommended to edit the file, but you can use it to view the default settings.

This file must be available and correctly formatted for the Connection Broker to start.

2. Next, the Connection Broker reads the global configuration file. The settings in the global configuration file override the default settings.

If the global configuration file is missing or malformed, the Connection Broker will start normally, and will read the user-specific configuration file, instead. A malformed global configuration file is ignored and the default settings or user-specific settings, if they exist, are used instead.

3. Last, the Connection Broker reads the user-specific configuration file, if it is available. The settings in the user-specific configuration file override the settings in the global configuration file, with the following exceptions:

- The following settings from the user-specific configuration are combined with the settings of the global configuration file:
  - In `general` element, the `key-stores`, `cert-validation` and `file-access-control` settings
  - In `profiles` element, all settings
  - In `static-tunnels` element, all settings.
- If a connection profile with the same name has been defined in both the global configuration file and user-specific configuration file, the latter one is used.
- If the `filter-engine` settings have been defined in the global configuration file, and the file is valid (not malformed), those settings are used, and any `filter-engine` settings made in the user-specific configuration file are ignored.

If the user-specific configuration file is missing, the Connection Broker will start using the previously read configuration files. However, if a user-specific configuration exists but is malformed, the Connection Broker will not start at all.

On Unix, the default configuration file locations are as follows:

- the default configuration:

```
/etc/ssh2/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-config-default.xml
```

- the global configuration: `/etc/ssh2/ssh-broker-config.xml`
- the user-specific configuration: `$HOME/.ssh2/ssh-broker-config.xml`
- the XML DTD:

```
/etc/ssh2/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd
```

On Windows, the default configuration file locations are as follows:

- the default configuration:

```
"C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml"
```

- the global configuration:

```
"C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Broker\ssh-broker-config.xml"
```

- the user-specific configuration: `"%APPDATA%\SSH\ssh-broker-config.xml"`

- the XML DTD:

```
"C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia AUX\ssh-broker-ng\ssh-broker-ng-config-1.dtd"
```

The following sections describe the options available in the Connection Broker configuration file. For more information on the syntax of the configuration file, see [Section A.4](#).

## Environment Variables

Two kinds of environment variables can be used in the Connection Broker configuration file. In addition to the system-level environment variables, you can use special variables that are SSH Tectia specific. The environment variables take precedence over the special variables. So if an environment variable and a special variable have the same name, the environment variable will be used.

All alphanumeric characters and the underscore '\_' sign are allowed in environment variables. The variable name ends to the first character that is not allowed.

You can define for example file or directory paths with environment variables, and they will be expanded to their values as explained below.

`%VARIABLENAME%`

Replaced with the value of the environment variable if one has been defined. The variable is matched case-insensitively. If the variable is not defined, the string '`%VARIABLENAME%`' is the result.

**\$VARIABLENAME**

Replaced with the value of the environment variable if one has been defined. The variable is matched case-sensitively on Unix and case-insensitively on Windows. If the variable is not defined, it is replaced with an empty string.

**\${VARIABLENAME}*text***

Replaced with the value defined for '\$VARIABLENAME' with the '*text*' appended to it.

**\${VARIABLENAME:-*default\_value*}**

Replaced with the value defined for '\$VARIABLENAME', or replaced with the '*default\_value*' if the variable is not set.

**The SSH Tectia specific special variables are:****%U or %username%**

Replaced with the currently logged in user name.

**%username-without-domain%**

Replaced with the currently logged in user name in short format, i.e. without the domain part. Available on Windows.

**%G or %groupname%**

Replaced with the group name of the currently logged in user.

**%D or %homedir%**

Replaced with the home directory defined for the currently logged in user.

**%IU or %userid%**

Replaced with the user identifier defined for the currently logged in user.

**%IG or %groupid%**

Replaced with the group identifier defined for the currently logged in user.

The special variables can also be entered using the Unix format, for example, \$username.

**Document Type Declaration and the Root Element**

The broker configuration file is a valid XML file and starts with the Document Type Declaration.

The root element in the configuration file is `secsh-broker`. It can include `general`, `default-settings`, `profiles`, `static-tunnels`, `gui`, `filter-engine`, and `logging` elements.

An example of an empty configuration file is shown below:

```
<!DOCTYPE secsh-broker SYSTEM "ssh-broker-ng-config-1.dtd">
<secsh-broker version="1.0">
  <general />
  <default-settings />
```

```
<profiles />
<static-tunnels />
<gui />
<filter-engine />
<logging />

</secsh-broker>
```

On SSH Tectia Client, the `filter-engine` element is used only when the optional transparent TCP tunneling feature has been installed and activated.

## The general Element

The `general` element contains settings such as the cryptographic library and the key stores to be used.

The `general` element can contain zero or one instance of the following elements: `crypto-lib`, `cert-validation`, `key-stores`, `user-config-directory`; and multiple `known-hosts` elements.

### `crypto-lib`

This element selects the cryptographic library mode to be used. Either the standard version (`standard`) or the FIPS 140-2 certified version (`fips`) of the cryptographic library can be used. The library name is given as a value of the `mode` attribute. By default, standard cryptographic libraries are used.

FIPS mode will be used if it is so specified either in the global or the user configuration file (or both).

```
<crypto-lib mode="standard" />
```

In the FIPS mode, the cryptographic operations are performed according to the rules of the FIPS 140-2 standard. The FIPS library includes the `3des-cbc`, `aes128-cbc`, `aes192-cbc`, and `aes256-cbc` ciphers and the `hmac-sha1` MAC.



### Note

Setting the FIPS mode does not prevent using algorithms from the crypto plugins. For example, CryptiCore can be used even when the main cryptographic library is set into the FIPS mode. To enforce that only FIPS-compliant algorithms are used, disable the non-FIPS algorithms from the configuration. See [cipher](#) and [mac](#).

For a list of platforms on which the FIPS library has been validated or tested, see *SSH Tectia Client/Server Product Description*.

### `cert-validation`

This element defines public-key infrastructure (PKI) settings used for validating remote server authentication certificates. The element can have the following attributes: `end-point-identity-check`, `default-domain`, `http-proxy-url`, and `socks-server-url`.

The `end-point-identity-check` attribute specifies whether the client will verify the server's hostname or IP address against the Subject Name or Subject Alternative Name (DNS Address) specified in the server host certificate. The default value is `yes`. If set to `no`, the fields in the server host certificate are *not* verified and the certificate is accepted based on the validity period and CRL check only.



## Caution

Setting `end-point-identity-check="no"` is a security risk. Then anyone with a certificate issued by the same trusted certification authority (CA) that issues the server host certificates can perform a man-in-the-middle attack on the server.

The `default-domain` attribute can be used when the end-point identity check is enabled. It specifies the default domain part of the remote system name and it is used if only the base part of the system name is available. The `default-domain` is appended to the system name if it does not contain a dot (`.`).

If the default domain is not specified, the end-point identity check fails, for example, when a user tries to connect to a host `"rock"` giving only the short hostname and the certificate contains the full DNS address `"rock.example.com"`.

The `http-proxy-url` attribute defines an HTTP proxy and the `socks-server-url` attribute defines a SOCKS server for making LDAP or OCSP queries for certificate validity.

The address of the server is given as the value of the attribute. The format of the address is `socks://username@socks_server:port/network/netmask,network/netmask ...` (with a SOCKS server) or `http://username@proxy_server:port/network/netmask,network/netmask ...` (with an HTTP proxy).

For example, to make the SOCKS server use host `socks.ssh.com` and port 1080 for connections outside of networks 192.196.0.0 (16-bit domain) and 10.100.23.0 (8-bit domain), and to get these networks connected directly, set `socks-server-url` as follows:

```
"socks://mylogin@socks.ssh.com:1080/192.196.0.0/16,10.100.23.0/24"
```

The `cert-validation` element can contain multiple `ldap-server`, `ocsp-responder`, `crl-prefetch` elements, one `dod-pki` element, and multiple `ca-certificate` and `key-store` elements. The elements have to be in the listed order.

### ldap-server

This element specifies an LDAP server address and port used for fetching CRLs and/or subordinate CA certificates based on the issuer name of the certificate being validated. Several LDAP servers can be specified by using several `ldap-server` elements.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be verified if the point exists.

The default value for port is 389.

### ocsp-responder

This element specifies an OCSP (Online Certificate Status Protocol) responder service address in URL format with attribute `url`. Several OCSP responders can be specified by using several `ocsp-responder` elements.

If the certificate has a valid Authority Info Access extension with an OCSP Responder URL, it will be used instead of this setting. Note that for the OCSP validation to succeed, both the end-entity certificate and the OCSP Responder certificate must be issued by the same CA.

The `validity-period` (in seconds) can be optionally defined. During this time, new OCSP queries for the same certificate are not made but the old result is used. The default validity period is 0 (a new query is made every time).

### crl-prefetch

This element instructs SSH Tectia Client to periodically download a CRL from the specified URL. The `url` value can be an LDAP or HTTP URL, or it can refer to a local file. The file format must be either binary DER or base64, PEM is not supported.

To download CRLs from the local file system, define the file URL in this format:

```
file:///absolute/path/name
```

To download CRLs from an LDAP server, define the LDAP URL in this format:

```
ldap://ldap.server.com:389/CN=Root%20CA,OU=certification  
%20authorities,DC=company,DC=com?certificaterevocationlist
```

Use the `interval` attribute to specify how often the CRL is downloaded. The default is 3600 seconds.

### dod-pki

This element defines whether the certificates are required to be compliant with the US Department of Defense Public-Key Infrastructure (DoD PKI). In practise, this means that the Digital Signature bit must be set in the Key Usage of the certificate. The `enable` attribute can have a value of `yes` or `no`. The default is `no`.

### ca-certificate

This element defines a certification authority (CA) used in server authentication. It can have four attributes: `name`, `file`, `disable-crls`, and `use-expired-crls`.

The `name` attribute must contain the name of the CA.

The element must either contain the path to the X.509 CA certificate file as a value of the `file` attribute, or include the certificate as a base64-encoded ASCII block.

CRL checking can be disabled by setting the `disable-crls` attribute to `yes`. The default is `no`.

Expired CRLs can be used by setting a numeric value (in seconds) for the `use-expired-crls` attribute. The default is 0 (do not use expired CRLs).

## key-store

This element defines CA certificates stored in an external key store for server authentication. Currently it is used only on z/OS for CA certificates stored in System Authorization Facility (SAF).

CRL checking can be disabled by setting the `disable-crls` attribute to `yes`. The default is `no`.

Expired CRLs can be used by setting a numeric value (in seconds) for the `use-expired-crls` attribute. The default is 0 (do not use expired CRLs).

An example of a certificate validation configuration is shown below:

```
<cert-validation end-point-identity-check="yes"
    default-domain="example.com"
    http-proxy-url="http://proxy.example.com:8080">
  <ldap-server address="ldap://ldap.example.com:389" />
  <ocsp-responder url="http://ocsp.example.com:8090" validity-period="0" />
  <crl-prefetch url="file:///full.path.to.crlfile" interval="1800" />
  <dod-pki enable="no" />
  <ca-certificate name="ssh_cal"
    file="ssh_cal.crt"
    disable-crls="no"
    use-expired-crls="100" />
</cert-validation>
```

## key-stores

This element defines settings for user public-key and certificate authentication.

Under the `<general>` element, there can be one `<key-stores>` instance which in turn can have any number of `<key-store>`, `<user-keys>`, and `<identification>` elements, and the order of the elements is free.

Special variables and environment variables can be used when defining the values for the elements. The following variables can be used and they will be expanded as follows:

- `%U = %USERNAME%` = user name
- `%USERNAME-WITHOUT-DOMAIN%` = user name without the domain part
- `%IU = %USERID%` = user ID (not on Windows)
- `%IG = %GROUPID%` = user group ID (not on Windows)
- `%D = %HOMEDIR%` = the user's home directory
- `%G = %GROUPNAME%` = the name of the user's default group

Also environment variables are replaced with their current values. For example it is possible to use strings `$HOME` or `%HOME%` to expand to user's home directory (if environment variable `HOME` is set).

## Note

Short alias names (for example, `%u`) are case-sensitive and long alias names (for example, `%USERNAME%`) are case-insensitive.

### key-store

Each of the `key-store` elements configures one key store provider. The `key-stores/key-store` element can take the following attributes: `type` and `init`.

The `type` attribute is the key store type. The currently supported types are "entrust", "mscapi", "pkcs11", "software", and "zos-saf". Entrust is supported on Windows, only.

The `init` attribute is the initialization info specific to the key-store-provider. The initialization string can contain special strings explained above in `key-stores`, see [key-stores](#).

For key store configuration examples, see [the section called "Key Store Configuration Examples"](#).

### user-keys

The `user-keys` element can be used to override the default directory for the user keys. The `user-keys` element can take the following attributes:

The `directory` attribute defines the directory where the user private keys are stored. Enter the full path.

The `passphrase-timeout` attribute defines the time (in seconds) after which the passphrase-protected private key will time out, and the user must enter the passphrase again. The default is 0, meaning that the passphrase does not time out. The value of this element should be longer than the `passphrase-idle-timeout` value.

By default, the Connection Broker keeps the passphrase-protected private keys open once the user has entered the passphrase successfully. This can be changed with the passphrase timeout options. When `passphrase-timeout` is set, the private key stays open (usable without further passphrase prompts) until the timeout expires. The `passphrase-timeout` attribute sets the hard timeout, that is set only once when the key is opened and will not be reset even if the key is used multiple times.

The `passphrase-idle-timeout` attribute defines the time (in seconds) after which the passphrase-protected private key will time out unless the user accesses or uses the key. The `passphrase-idle-timeout` is reset every time the key is accessed. The default is 0, meaning that the passphrase never times out.

Both of the timeout options can be set simultaneously, but notice that if the idle timeout is set longer than the hard timeout, the idle timeout has no effect.



## identification

The `identification` element can be used to override the default location of the identification file that defines the user keys. The `identification` element can take the following attributes:

The `file` attribute specifies the location of the identification file. Enter the full path.

The `base-path` attribute defines the directory where the identification file expects the user private keys to be stored. This element can be used to override the default relative path interpretation of the identification file (paths relative to the identification file directory).

The `passphrase-timeout` attribute defines the time (in seconds) after which the user must enter the passphrase again. The default is 0, meaning that the passphrase is not re-requested.

The `passphrase-idle-timeout` attribute defines a time (in seconds) after which the passphrase times out if there are no user actions. The default is 0, meaning that the passphrase does not time out.

The timeout settings affect only those private keys that are listed in the identification file.

## strict-host-key-checking

### Note

This element is deprecated starting from SSH Tectia Client version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `host-key-always-ask` and `accept-unknown-host-keys` options. See [auth-server-publickey](#) for details.

## host-key-always-ask

### Note

This element is deprecated starting from SSH Tectia Client version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `strict-host-key-checking` and `accept-unknown-host-keys` options. See [auth-server-publickey](#) for details.

## accept-unknown-host-keys

### Note

This element is deprecated starting from SSH Tectia Client version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `strict-host-key-checking` and `host-key-always-ask` options. See [auth-server-publickey](#) for details.

### Caution

Consider carefully before enabling this option. Disabling the host-key checks makes you vulnerable to man-in-the-middle attacks.

## user-config-directory

This element can be used to change the storage location of the user-specific configuration files away from the default which is `$HOME/.ssh2/` on Unix, and `"%APPDATA%\SSH"` on Windows. It can be used for example, if you want to store all client-side configurations to a centralized location.

When this element is added to the global configuration file, the Connection Broker reads the following user-specific files in the defined location:

- user's key file
- user's own configuration files
- user's known host keys
- user's random\_seed file
- Windows GUI profile files: 1.ssh2, 2.ssh2

### Note

Stop all existing SSH applications before modifying the `<user-config-directory>` setting in the Connection Broker configuration.

The syntax is:

```
<user-config-directory path="%variable%/dirname" />
```

The value can be defined with the following variables:

- `%U`: The user name.

- `%username%`: The user name.
- `%username-without-domain%`: The user name without domain definition.
- `%D:` The user's home directory.
- `%homedir%`: The user's home directory.
- `%USER_CONFIG_DIRECTORY%`: The user-specific configuration directory.
- `%IU:` The user's ID, on Unix only
- `%userid%`: The user's ID, on Unix only
- `%IG:` The group ID, on Unix only
- `%groupid%`: The group ID, on Unix only

The default is `%USER_CONFIG_DIRECTORY%`. This variable can be used in this configuration setting to refer to the user-specific configuration directory: `$HOME/.ssh2` on Unix, and `%APPDATA%\SSH` on Windows. This variable cannot be used in other settings.

### file-access-control

On Unix, this element can be used to enable checking of file access permissions defined for the global and user-specific configuration files, and for the private keys files. If the permissions are not as expected, the Connection Broker will refuse to start, or to use certain private keys.

By default this setting is disabled. On Windows, this element has no effect.

The file permissions are checked differently, if the `file-access-control` element is set in both the global and user configuration files, or just in one of them. See the following table for details:

**Table A.1. Different file-access-control effects**

Setting in:		Permissions checked in:		
Global config	User config	Global config	User config	Private key files
yes	yes	Checked	Checked	Checked
yes	-	Checked	Checked	Checked
-	yes	Not checked	Checked	Checked
yes	no	Checked	Checked	Not checked
no	yes	Not checked	Checked	Checked
no / -	no / -	no checking	no checking	no checking

In the table: No means `file-access-control enable="no"`. Sign - means that the setting is not included in the file at all.

When the file access permissions are checked, the controls are applied as follows:

- Expected permissions for the global configuration file: read rights for all, write rights only for the user and group. If the permissions are any wider, the Connection Broker will not start.
- Expected permissions for the user configuration file: only the user has read and write rights. If the permissions are any wider, the Connection Broker will not start.
- Expected permissions for the private key files: only the user has read and write rights. If the permissions are any wider, keys that do not pass the check will be ignored.

### known-hosts

This element can be used to specify locations for storing the host keys of known server hosts, and to define the storage format of the host key files. If no `known-hosts` directories are specified, the known host keys are stored to the default directories. See [the section called “Files”](#) for the default locations. On z/OS (only), this element can contain `key-store` elements.

This element can be used:

- To specify non-default directories that contain the public-key data or public-key files of known server hosts.
- To specify a non-default location for OpenSSH-style `known_hosts` files that contain the public-key data of known server hosts.
- (*On z/OS*) To specify a SAF key store that contains the certificates of known server hosts.

The server host keys are searched in the `known-hosts` paths in the order they are specified in the configuration. The settings of the last defined `known-hosts` element are used when storing new host keys.

If you define any `known-hosts` file settings, the default OpenSSH files will be overridden. So if you wish to make the Connection Broker use both the default OpenSSH locations and other locations specified in the configuration, you need to specify all the locations separately.

You can define several `known-hosts` elements, and each of them can contain one or several attributes: `path`, `directory`, `file` and `filename-format`.

The `path` attribute requires a full path to the `known-hosts` file or directory as the value. For example:

```
<known-hosts path="/u/username/.ssh/known_hosts" />
<known-hosts path="/etc/ssh2/hostkeys" />
<known-hosts path="/u/username/.ssh2/hostkeys" />
<known-hosts path="/h/username/hostkeys" filename-format="plain" />
```

The `directory` attribute is used to define that known host keys are saved to a non-default directory. Enter the complete path to the directory as the value. If the defined directory does not exist, it will be created during the first connection attempt. If a file is found in its place, the connection will be made but

the host key will not be stored, and the user gets a warning about it. The `filename-format` attribute can be used together with the `directory` setting to define in which format the host key files will be stored. Example of the `directory` attribute:

```
<known-hosts directory="<path_to_dir>/MyKEYS"
  filename-format="plain" />
```

The path or directory (whichever is present) defined in the last `known-hosts` element in the configuration file will be used when storing new known host keys. If both attributes are present in the last `known-hosts` element, the location specified in the `directory` attribute will be used.

The `file` attribute is used to point to an OpenSSH-style `known_hosts` file. Enter the complete path to the file as the value. If a directory is found in its place, it is considered an error, and the connection attempt will fail. In case the `known-hosts` element only contains the `file` attribute, and the defined OpenSSH `known_hosts` file exists, the received host keys are searched first in the defined file, and if not found there, the search continues in the default Tectia-specific locations.

Example of the `file` attribute:

```
<known-hosts file="<path_to_file>/ssh2/openSSH_keys" />
```

An empty `file` or `path` attribute will disable the handling of the OpenSSH `known_hosts` file:

```
<known-hosts file="" />
or
<known-hosts path="" />
```

The `filename-format` attribute defines the format in which new host key files are stored. The `filename-format` attribute is only relevant for the last specified `known-hosts` element and for the default directory.

The `filename-format` attribute takes the values: `hash` (default), `plain`, and `default` (equals to `hash`).

With value `hash`, the host key files will be stored in format: `keys_<hash>`, for example `"keys_182166d2efe5a134d3fb948646e0b48f780bff6c"`.

With value `plain`, the file name format will be `key_<port>_<hostname>.pub`, where `<port>` is the port the Secure Shell server is running on and `<host>` is the hostname you use when connecting to the server; for example `"key_22_my.example.com.pub"`.

Setting `<known-hosts filename-format="plain" />` changes the storage format of host key files for the next `known-hosts` elements or for the default storage location if no other `known-hosts` elements are present.

The `filename-format="default"` alternative can be used as the last option when the same `known-hosts` element is used to define several locations for the host keys some of which store the keys in plain format.

For more information on the host key storage formats, see [Section 5.1.1](#).

## key-store

This element defines an external key store for certificates of known server hosts. Currently it is used only on z/OS for server certificates stored in System Authorization Facility (SAF).

## extended

This element is reserved for future use.

## Key Store Configuration Examples

### Example with Software Provider

The software provider handles key pairs stored on disk in standard Secure Shell v2 or legacy OpenSSH formats and X.509 certificates stored in native X.509, PKCS#7, and PKCS#12 formats.

To add a single key file (for example, `/u/exa/keys/enigma` and `/etc/my_key`), specify both the private key file and the public key file:

```
<key-stores>
  <key-store type="software"
    init="key_files(/u/exa/keys/enigma.pub,/u/exa/keys/enigma)" />
  <key-store type="software"
    init="key_files(/etc/my_key.pub,/etc/my_key)" />
</key-stores>
```

To add all keys from a specific directory (for example all keys from `/u/exa/keys` and `/etc/keys`):

```
<key-stores>
  <key-store type="software"
    init="directory(path(/u/exa/keys))" />
  <key-store type="software"
    init="directory(path(/etc/keys))" />
</key-stores>
```

### Example with Entrust Provider

The Entrust provider handles keys and certificates stored in the proprietary Entrust format. You should provide the initialization file and the profile-specific file for the Entrust provider. For example:

```
<key-stores>
  <key-store type="entrust"
    init="ini-file(/etc/entrust.ini),profile-file(/etc/profile.epf)" />
</key-stores>
```

### Example with PKCS#11 Provider

The PKCS#11 provider handles keys and certificates stored in PKCS#11 tokens (for example, smart cards or USB tokens).

Specify the dynamic library path for the PKCS provider and all or a specific slot. For example, with all slots:

```
<key-stores>
  <key-store type="pkcs11" init="dll(/usr/lib/pkcs.so),slots(all)" />
</key-stores>
```

For example, with one slot named `sesam`:

```
<key-stores>
  <key-store type="pkcs11" init="dll(/usr/local/lib/pkcs.so),slots(sesam)" />
</key-stores>
```

## The default-settings Element

The `default-settings` element defines the default connection-related settings. Profile-specific settings can override these settings. See [the section called “The profiles Element”](#).

The `default-settings` element can contain zero or one instance of the following elements in the listed order: `ciphers`, `macs`, `transport-distribution`, `rekey`, `authentication-methods`, `hostbased-default-domain`, `compression`, `proxy`, `idle-timeout`, `tcp-connect-timeout`, `keepalive-interval`, `exclusive-connection`, `server-banners`, `forwards`, `extended`, `remote-environment`, `server-authentication-methods`, `authentication-success-message`, and `sftp3-mode`.

### ciphers

This element defines the ciphers that the client will propose to the server. The `ciphers` element can contain multiple `cipher` elements.

The ciphers are tried in the order they are specified.

With SSH Tectia Server for Linux on IBM System z, the client tools will automatically use hardware acceleration (CPACF), if it is available, on cryptographic operations with the 3DES and AES algorithms.

### cipher

This element selects a cipher name that the client requests for data encryption.

The supported ciphers are `3des-cbc`, `aes128-cbc`, `aes192-cbc`, `aes256-cbc`, `aes128-ctr`, `aes192-ctr`, `aes256-ctr`, `arcfour`, `blowfish-cbc`, `twofish-cbc`, `twofish128-cbc`, `twofish192-cbc`, `twofish256-cbc`, `crypticore128@ssh.com`, `seed-cbc@ssh.com`, and `none` (no encryption).

The default ciphers used by the Connection Broker are, in order: `crypticore128@ssh.com` (on Windows and Linux x86), `aes128-cbc`, `aes192-cbc`, `aes256-cbc`, `aes128-ctr`, `aes192-ctr`, `aes256-ctr`, `3des`, and `seed-cbc@ssh.com`.

The ciphers that can operate in the FIPS mode are `aes128-cbc`, `aes192-cbc`, `aes256-cbc`, and `3des-cbc`.

```
<ciphers>
  <cipher name="aes128-cbc" />
```

```
<cipher name="3des-cbc" />
</ciphers>
```

## macs

This element defines the MACs that the client will propose to the server. The `macs` element can contain multiple `mac` elements.

With SSH Tectia Server for Linux on IBM System z, the client tools will automatically use hardware acceleration (CPACF), if it is available, on cryptographic operations with the HMAC-SHA1 algorithms.

The MACs are tried in the order they are specified.

## mac

This element selects a MAC name that the client requests for data integrity verification.

The supported MAC algorithms are `hmac-md5`, `hmac-md5-96`, `hmac-sha1`, `hmac-sha1-96`, `crypticore-mac@ssh.com`, and `none` (no data integrity verification).

The default MACs used by the Connection Broker are, in order: `crypticore-mac@ssh.com` (on Windows and Linux x86), `hmac-md5`, and `hmac-sha1`.

The `hmac-sha1` algorithm can operate in the FIPS mode.

```
<macs>
  <mac name="hmac-sha1" />
</macs>
```

## transport-distribution

This setting defines the number of transport channels used by the Secure Shell connection. Using more than one transport may increase the throughput over low bandwidth connections.

The number of transports is given as value of the `num-transports` attribute. Currently, a value of 1 to 8 transports is supported. On Unix, the default is 1 transport. On Windows, the default is 2 transports.

```
<transport-distribution num-transports="1" />
```

## rekey

This element specifies the number of transferred bytes after which the key exchange is done again. The value "0" turns rekey requests off. This does not prevent the server from requesting rekeys, however. The default is 1000000000 (1 GB).

```
<rekey bytes="1000000000" />
```

## authentication-methods

This element specifies the authentication methods that are requested by the client-side components. The `authentication-methods` element can contain one of each: `auth-hostbased`, `auth-password`, `auth-`



publickey, auth-gssapi, and auth-keyboard-interactive. Alternatively, you can specify multiple authentication-method elements. The order of these elements is free.

The authentication methods are tried in the order the auth-\* or authentication-method elements are listed. This means that the least interactive methods should be placed first.

### **authentication-method**

This element specifies an authentication method name. It is included for backwards compatibility. Use the auth-\* elements instead.

The allowed authentication method names are: gssapi-with-mic, publickey, keyboard-interactive, password, and hostbased.

SSH Tectia Client supports host-based authentication only on Unix platforms.

```
<authentication-methods>
  <authentication-method name="hostbased" />
  <authentication-method name="gssapi-with-mic" />
  <authentication-method name="publickey" />
  <authentication-method name="keyboard-interactive" />
  <authentication-method name="password" />
</authentication-methods>
```

### **auth-hostbased**

This element specifies that host-based authentication will be used.

The auth-hostbased element can include a local-hostname element.

#### **local-hostname**

This element specifies the local hostname, as the value of the name attribute, that is advertised to the remote server during host-based authentication.

The remote server can use the client host name as a hint when locating the public key for the client host. This information is not significant to the authentication result, but makes it faster to find the relevant client host key, if the server has such a big storage of host identities, that trying them all would be infeasible.

### **auth-password**

This element specifies that password authentication will be used.

### **auth-publickey**

This element specifies that public-key authentication will be used.

The auth-publickey element can include a key-selection element.

### key-selection

This element specifies the key selection policy the client uses when proposing user public keys to the server. The `policy` attribute can take the values `automatic` (default) and `interactive-shy`.

In the `automatic` mode, the client tries keys in the following order:

1. Keys with public key available and private key without a passphrase (no user interaction)
2. Keys with public key available but private key behind a passphrase (one passphrase query)
3. Keys that need a passphrase to get the public key but private key without passphrase (one user query for each key which is considered and proposed to server, but no user interaction for actual public-key login)
4. The rest of the keys, that is, keys that need a passphrase to get the public key and also to get the private key

In the `interactive-shy` mode, the client does not try any keys automatically, but it prompts the user to select the key from a list of available keys. If the authentication with the selected key fails, the client will prompt the user again, removing the already tried key(s) from the list. If there is only one key candidate available, the key will be tried automatically without asking the user.

The `key-selection` element can include a `public-key` element.

### public-key

This element can be used to specify that only plain public keys or only certificates are tried during public-key authentication. The `type` attribute can take the values `plain` and `certificate`. The default is to try both plain public keys and certificates.

### auth-keyboard-interactive

This element specifies that keyboard-interactive methods will be used in authentication.

### auth-gssapi

This element specifies that GSSAPI will be used in authentication.

An example of authentication-methods configuration is shown below:

```
<authentication-methods>
  <auth-hostbased>
    <local-hostname name="host.example.com" />
  </auth-hostbased>
  <auth-gssapi />
  <auth-publickey>
```

```

    <key-selection policy="interactive-shy">
      <public-key type="plain" />
    </key-selection>
  </auth-publickey>
  <auth-keyboard-interactive />
  <auth-password>
    <password file="/path/filename" />
  </auth-password>
</authentication-methods>

```

### hostbased-default-domain

This element specifies the host's default domain name (as `name`). This element is used to make sure the fully qualified domain name (FQDN) of the client host is transmitted to the server when using host-based user authentication.

The default domain name is appended to the short hostname before transmitting it to the server. This is needed because some platforms (Solaris for instance) use the short format of the hostname, and with that the signature cannot be created.

The allowed formats of the default domain names are: `.example.com` and `example.com` (without the leading dot). For example:

```
<hostbased-default-domain name=".example.com" />
```

### compression

This element specifies whether to use compression on all traffic. When activated, the compression is applied to all transferred data on-the-fly.

The name of the compression algorithm and the compression level can be given as attributes. The `name` attribute can be defined as `none` (compression not used) or `zlib`, currently the only supported algorithm. By default, compression is not used.

The `level` attribute can be given an integer from 0 to 9. The default compression level is 6, when compression is activated but no level is given.

Example: to activate compression on the maximum level, make the following setting:

```
<compression name="zlib" level="9" />
```

Compression can also be activated per connection with command line tools. For information, see the [sshg3\(1\)](#), [sftpg3\(1\)](#) and [scpg3\(1\)](#) man pages.

### proxy

This element defines rules for HTTP proxy or SOCKS servers the client will use for connections. It has a single attribute: `ruleset`.

The format of the attribute value is a sequence of rules delimited by semicolons (;). Each rule has a format that resembles the URL format. In a rule, the connection type is given first. The type can be `direct`, `socks`, `socks4`, `socks5`, or `http-connect` (`socks` is a synonym for `socks4`). This is followed by the server address and port. If the port is not given, the default ports 1080 for SOCKS and 80 for HTTP are used.

After the address, zero or more conditions delimited by commas (,) are given. The conditions can specify IP addresses or DNS names.

```
direct:///[cond[,cond]...]
socks://server/[cond[,cond]...]
socks4://server/[cond[,cond]...]
socks5://server/[cond[,cond]...]
http-connect://server/[cond[,cond]...]
```

The IP address/port conditions have an address pattern and an optional port range:

```
ip_pattern[:port_range]
```

The `ip_pattern` may have one of the following forms:

- a single IP address `x.x.x.x`
- an IP address range of the form `x.x.x.x-y.y.y.y`
- an IP sub-network mask of the form `x.x.x.x/y`

The DNS name conditions consist of a hostname which may be a regular expression containing the characters "\*" and "?" and a port range:

```
name_pattern[:port_range]
```

An example `proxy` element is shown below. It causes the server to access the callback address and the `ssh.com` domain directly, access `*.example` with HTTP CONNECT, and all other destinations with SOCKS4.

```
<proxy ruleset="direct:///127.0.0.0/8,*.ssh.com;
    http-connect://http-proxy.ssh.com:8080/*.example;
    socks://fw.ssh.com:1080/" />
```

### idle-timeout

This element specifies how long idle time (after all connection channels are closed) is allowed for a connection before automatically closing the connection. The `time` is given in seconds. The `type` is always `connection`.

The default setting is 5 seconds. Setting a longer time allows the connection to the server to remain open even after a session (for example, `sshg3`) is closed. During this time, a new session to the server can be initiated without re-authentication. Setting the time to 0 (zero) terminates the connection immediately when the last channel to the server is closed.

```
<idle-timeout time="5" />
```

### tcp-connect-timeout

This element specifies a timeout for the TCP connection. When this setting is made, connection attempts to an Secure Shell server are stopped after the defined time if the remote host is down or unreachable. This timeout overrides the default system TCP timeout, and this timeout setting can be overridden by defining a `tcp-connect-timeout` setting per connection profile (in the `profiles` settings) or per connection (on command line).

The `time` is given in seconds. The factory default is 5 seconds. Value 0 (zero) disables this feature and the default system TCP timeout will be used.

```
<tcp-connect-timeout time="5" />
```

### keepalive-interval

This element specifies an interval for sending keepalive messages to the Secure Shell server. The `time` value is given in seconds. The default setting is 0, meaning that the keepalive messages are disabled.

```
<keepalive-interval time="0" />
```

### exclusive-connection

The `exclusive-connection` element can be used to specify that a new connection is opened for each new channel.

The word `yes` or `no` is given as the value of the `enable` attribute. The default is `no` (open connections are reused for new channels requested by a client).

### server-banners

This element defines whether the server banner message file (if it exists) is visible to the user before login. The word `yes` or `no` is given as the value of the `visible` attribute. The default is `yes`.

To eliminate server banners:

```
<server-banners visible="no" />
```

### forwards

This element contains `forward` elements that define whether X11 or agent forwarding (tunneling) are allowed on the client side.

#### forward

This element defines X11 or agent forwarding settings.

The `type` attribute defines the forwarding type (either `x11` or `agent`). The `state` attribute sets the forwarding on, off, or denied. If the forwarding is set as denied, the user cannot enable it on the command-line.

An example forward configuration, which denies X11 forwarding and allows agent forwarding globally, is shown below:

```
<forwards>
  <forward type="x11" state="denied" />
  <forward type="agent" state="on" />
</forwards>
```

For more information on using X11 and agent forwarding, see [Section 7.3](#) and [Section 7.4](#).

### extended

This element is reserved for future use.

### server-authentication-methods

This `server-authentication-methods` element can be used to force the Connection Broker to use only certain methods in server authentication. This element can contain `auth-server-publickey` and `auth-server-certificate` elements (one of each). Alternatively, you can specify up to two `authentication-method` elements. The order of these elements is free.

If only `auth-server-certificate` is specified, server certificate is needed. If no server certificate is received, connection fails.

If only `auth-server-publickey` is specified, (plain) server public key is needed. If no server public key is received, connection fails.

If both `auth-server-certificate` and `auth-server-publickey` are specified, server certificate is used if present. Otherwise server public key is used.

### auth-server-certificate

The `auth-server-certificate` element specifies that certificates are used for server authentication.

### auth-server-publickey

The `auth-server-publickey` element specifies that public host keys are used for server authentication.



### Note

The host key policy settings have changed in version 6.1.4 and are now defined in the `auth-server-publickey` element.

The element takes attribute `policy` that defines how unknown server host keys are handled. It can have the following values:

- **strict**: Connect to the server only if the host key is found from the host key store and matches.

If the policy is set to **strict**, the Connection Broker never adds host keys to the user's `.ssh2/hostkeys` directory upon connection, and refuses to connect to hosts whose key has changed. This provides maximum protection against man-in-the-middle attacks. However, it also means you must always obtain host keys via out-of-band means, which can be troublesome if you frequently connect to new hosts.

- **ask** (default): If the server host key is not found from the host key store, the user will be asked if he wants to accept the host key. If the host key has changed, the user is warned about it and asked how to proceed. If the client application is not able to ask the user (for example, **sftpg3** in batch mode, `-B`), the connection will be disconnected.
- **trust-on-first-use** or **tofu**: If the server host key is not found, it is stored to the user's `.ssh2/hostkeys` directory. If the host key has changed, the connection will be disconnected.
- **advisory**: Use of this setting effectively disables server authentication, which makes the connection vulnerable to active attackers.

If the server host key is not found in the host key store, it will be added to the user's `.ssh2/hostkeys` directory without user interaction. If the host key has changed, the connection will be continued without user interaction. The incident will be audited if logging is enabled.

When the policy is set to **advisory**, the keys from new hosts are automatically accepted and stored to the host key database without prompting acceptance from the user. However, changed host keys (from hosts whose keys are already in the database) are not stored, but they are accepted for that connection only. This has the same effect as automatically answering `Once` to all accept-host-key prompts.

This setting should be used only if logging is enabled for the Connection Broker (by default, logging is enabled only if the Broker is run by the MFT Events service).



## Caution

Consider carefully before setting the policy to **advisory**. Disabling the host-key checks makes you vulnerable to man-in-the-middle attacks.

In policy modes other than **strict**, if logging is enabled for the Connection Broker, SSH Tectia Client will log information about changed and new host public keys with their fingerprints in the `syslog` (on Unix) or Event Viewer (on Windows).



## Note

When transparent FTP tunneling or FTP-SFTP conversion is used, accepting the host key cannot be prompted from the user. Either the policy must be set to **tofu** or the host keys

of the Secure Shell tunneling and SFTP servers must be obtained beforehand and stored based on the IP addresses of the servers.

If the `policy` attribute is not defined, the host key policy is interpreted based on the values of the old `strict-host-key-checking`, `host-key-always-ask`, and `accept-unknown-host-keys` options as shown in [Table A.2](#) below.



## Note

In version 6.1.4 and later, the host key policy setting in the user-specific configuration file always takes precedence over the setting in the global configuration file.

**Table A.2. Interpretation of old host key policy (SSH Tectia Client 5.0.0-6.1.3) to new host key policy (SSH Tectia Client 6.1.4 and later)**

<b>strict-host-key-checking</b>	<b>accept-unknown-host-keys</b>	<b>host-key-always-ask</b>	<b>Policy</b>
-	-	-	<b>ask (default)</b>
enabled	-	-	<b>strict</b>
enabled	enabled	-	<b>strict</b>
enabled	enabled	enabled	<b>ask</b>
enabled	-	enabled	<b>ask</b>
-	enabled	-	<b>trust on first use</b>
-	enabled	enabled	<b>ask</b>
-	-	enabled	<b>ask</b>

## authentication-method

The `server-authentication-methods/authentication-method` element specifies an authentication method name. This element is included for backwards compatibility. Use the `auth-server-*` elements instead.

```
<server-authentication-methods>
  <authentication-method name="publickey" />
  <authentication-method name="certificate" />
</server-authentication-methods>
```

An example `server-authentication-methods` element is shown below:

```
<server-authentication-methods>
  <auth-server-publickey policy="ask" />
  <auth-server-certificate />
</server-authentication-methods>
```



### authentication-success-message

This setting defines whether the `AuthenticationSuccessMsg` messages are output. The `authentication-success-message` element takes attribute `enable` with value `yes` or `no`. The default is `yes`, meaning that the messages are output and logged.

### sftp3-mode

This setting defines how the **sftp3** client behaves when transferring files. The `sftp3-mode` element takes attribute `compatibility-mode` with the following values:

- `tectia` (the default) - **sftp3** transfers files recursively, meaning that files from the current directory and all its subdirectories are transferred.
- `ftp` - the `get/put` commands are executed as `sget/sput` meaning that they transfer a single file; and commands `mget/mput` have recursion depth set to 1 meaning that they only transfer files from the specified directory, not from subdirectories.
- `openssh` - commands `get/put/mget/mput` behave alike, and the recursion depth is set to 1, meaning that only files from the specified directory are transferred, not from subdirectories.

The recursion depth can be overridden by using the **sftp3** client's commands `get/put/mget/mput` with command-line option `--max-depth="LEVEL"`. For more information, see [sftp3\(1\)](#).

### remote-environment

This element contains `environment` elements which define the environment variables to be passed to the server from the client side. The environment variables are then set on the server when requesting a command, shell or subsystem.

Note that the server can restrict the setting of environment variables.

#### environment

This element defines the name and value of the environment variables, and whether the Connection Broker should process the value. Possible attributes are `name`, `value`, and `format`.

An example remote environment configuration:

```
<remote-environment>
  <environment name="FOO" value="bar" />
  <environment name="QUX" value="%Ubaz" format="yes" />
  <environment name="ZAPPA" value="%Ubaz" />
</remote-environment>
```

You can use `%U` in the `value` to indicate a user name. When `format="yes"` is also defined, the Connection Broker processes the `%U` into the actual user name before sending it to the server.

Let's assume the user name is `joedoe` in this example. The example configuration results in the following environment variables on the server side, provided that the server allows setting the environment variables:

```
FOO=bar
QUX=joedoebaz
ZAPPA=%Ubaz
```

You can override the remote environment settings made in the configuration file if you use the `sshg3` command with the following arguments on the command-line client: `--remote-environment` or `--remote-environment-format`

For information on the command-line options, see [sshg3\(1\)](#).

## The profiles Element

The `profiles` element defines the connection profiles for connecting to the specified servers. Element `profiles` can contain multiple `profile` elements. Each profile defines the connection rules to one server. The settings in the `profile` element override the default connection settings.

When a profile is used for the connection, the settings in the profile override the default settings. See [the section called “The default-settings Element”](#).

### profile

The `profile` element defines a connection profile. It has the following attributes: `id`, `name`, `host`, `port`, `protocol`, `connect-on-startup`, `user`, and `gateway-profile`.

The profile `id` must be a unique identifier that does not change during the lifetime of the profile.

An additional `name` can be given to the profile. This is a free-form text string. The name can be used for connecting with the profile on the command line, so define a unique name for each profile.

The `host` attribute defines the address of the Secure Shell server host and it is a mandatory setting. The address can be either an IP address or a domain name. The value `host="*"` can be used to prompt the user to enter the host address when starting the session.

An empty value `host=""` can be used when the profile is used with transparent TCP or FTP tunneling or FTP-SFTP conversion and the host name is taken from the application (`filter-engine/rule[@host-name-from-app="yes"]`). See [rule](#) for details.

The `port` is a mandatory setting. It defines the port number of the Secure Shell server listener. The default port is 22.

The `protocol` is a mandatory setting. It defines the used communications protocol. Currently the only allowed value is `secsh2`.

If you want to make the connection specified by the profile automatically when the Connection Broker is started, set the value of the `connect-on-startup` attribute to `yes`. In this case, give also the `user` attribute (the username the connection is made with). You also need to set up some form of non-interactive authentication for the connection.

The `user` attribute specifies the user name for opening the connection. The value `%USERNAME%` can be used to set the user name to the current user. The value `user=" * "` can be used to prompt the user to enter the user name when logging in.

The `gateway-profile` attribute can be used to create nested tunnels. The tunnels defined under the `local-tunnel` element of the profile, and the tunnels defined under `filter-engine` and `static-tunnels` that refer to the profile can be nested. The profile name through which the connection is made is given as the value of the attribute. The first tunnel is created using the gateway host profile and from there the second tunnel is created to the host defined in this profile.

### **hostkey**

This element gives the path to the remote server host public key file as a value of the `file` attribute.

Alternatively, the public key can be included as a base64-encoded ASCII block.

### **ciphers**

This element defines the ciphers used with this profile. See [ciphers](#) for details.

### **macs**

This element defines the MACs used with this profile. See [macs](#) for details.

### **transport-distribution**

This element defines the transport distribution for this profile. See [transport-distribution](#) for details.

### **rekey**

This element defines the rekeying settings used with this profile. See [rekey](#) for details.

### **authentication-methods**

This element defines the authentication methods used with this profile. See [authentication-methods](#) for details.

### **user-identities**

This element specifies the identities used in user public-key authentication. In contrast to the `key-stores` element that specifies all the keys that are available for the Connection Broker, this element can be used to control the keys that are attempted in authentication when this connection profile is used and to specify the order in which they are attempted.

The `user-identities` element can contain multiple `identity` elements. When multiple `identity` elements are used, they are tried out in the order they are listed.

The `identity` element has the following attributes: `identity-file`, `file`, `hash`, `id`, and `data`.

The `identity-file` attribute specifies that the user identity is read in the identification file used with public-key authentication. Enter the full path to the file if it is located somewhere else than the default identification file directory which is `$HOME/.ssh2`. See also [ssh-broker-g3\(1\)](#).

The `file` attribute specifies the path to the public-key file (primarily) or to a certificate. Enter the full path and file name as the value.

The `hash` attribute is used to enter the hash of the public key that will be used to identify the related private key. The key must be available for the Connection Broker. The public key hashes of the available keys can be listed with the **ssh-broker-ctl** tool. See also [ssh-broker-ctl\(1\)](#).

The `id` attribute is reserved for future use.

The `data` attribute is reserved for future use.

An example `user-identities` element is shown below:

```
<user-identities>
  <identity identity-file="C:\\ mykey" />
  <identity file="$HOME/user/.ssh2/id_dsa_2048_a" />
  <identity file="C:\\private_keys\\id_dsa_2048_a" />
  <identity hash="#a8edd3845005931aaa658b5573609e7d31e23afd" />
</user-identities>
```

### compression

This element defines the compression settings used with this profile. See [compression](#) for details.

### proxy

This element defines the HTTP proxy and SOCKS server settings used with this profile. See [proxy](#) for details.

If `gateway-profile` has been defined for this profile, the proxy setting is ignored and the default proxy setting or the proxy setting of the gateway profile is used instead.

### idle-timeout

This element defines the idle timeout settings used with this profile. See [idle-timeout](#) for details.

### **tcp-connect-timeout**

This element defines the TCP connection timeout for this profile. The timeout is used to terminate connection attempts to Secure Shell servers that are down or unreachable. The default value is 5 seconds. See [tcp-connect-timeout](#) for details.

### **keepalive-interval**

This element defines an interval for sending keepalive messages to the Secure Shell server. The setting applies to this profile. The default value is 0, meaning that no keepalive messages are sent. See [keepalive-interval](#) for details.

### **exclusive-connection**

This element defines whether a new connection is opened for each new channel when a connection is made with this profile. See [exclusive-connection](#) for details.

### **server-banners**

This element defines the server banner setting used with this profile. See [server-banners](#) for details.

### **forwards**

This element defines the forwards allowed with this profile. See [forwards](#) for details.

### **tunnels**

The `tunnels` element defines the tunnels that are opened when a connection with this profile is made. The element can contain multiple `local-tunnel` and `remote-tunnel` elements.

#### **local-tunnel**

This element defines a local tunnel (port forwarding) that is opened automatically when a connection is made with the connection profile. It has five attributes: `type`, `listen-port`, `listen-address`, `dst-host`, `dst-port`, and `allow-relay`.

The `type` attribute defines the type of the tunnel. This can be `tcp` (default, no special processing), `ftp` (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session), or `socks` (SSH Tectia Client/ConnectSecure will act as a SOCKS server for other applications, creating forwards as requested by the SOCKS transaction).

The `listen-port` attribute defines the listener port number on the local client.

The `listen-address` attribute can be used to define which network interfaces on the client should be listened. Its value can be an IP address belonging to an interface on the local host. Value `0.0.0.0` listens to all interfaces. The default is `127.0.0.1` (localhost loopback address on the client). Setting any other value requires setting `allow-relay="yes"`.

Whenever a connection is made to the specified listener, the connection is tunneled over Secure Shell to the remote server and another connection is made from the server to a specified destination host and port (`dst-host`, `dst-port`). The connection from the server onwards will not be secure, it is a normal TCP connection.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = server host).

The `allow-relay` attribute defines whether connections to the listened port are allowed from outside the client host. The default is `no`. If you use `allow-relay="yes"`, it will check also the `listen-address` setting.

For more information on using local tunnels, see [Section 7.1](#).

### **remote-tunnel**

This element defines a remote tunnel (port forwarding) that is opened automatically when a connection is made with the connection profile. It has four attributes: `type`, `listen-port`, `listen-address`, `dst-host`, `dst-port`, and `allow-relay`.

The `type` attribute defines the type of the tunnel. This can be either `tcp` (default, no special processing) or `ftp` (temporary forwarding is created for FTP data channels, effectively securing the FTP session between the client and server).

The `listen-port` attribute defines the listener port number on the remote server.

The `listen-address` attribute can be used to define which network interfaces on the server should be listened. Its value can be an IP address belonging to an interface on the server host. Value `0.0.0.0` listens to all interfaces. The default is `127.0.0.1` (localhost loopback address on the server). Setting any other value requires that `allow-relay="yes"`.

Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified destination host and port (`dst-host`, `dst-port`). The connection from the client onwards will not be secure, it is a normal TCP connection.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = client host).

The `allow-relay` attribute defines whether connections to the listener port are allowed from outside the server host. The default is `no`.

For more information on using remote tunnels, see [Section 7.2](#).

**extended**

This element is reserved for future use.

**remote-environment**

This element defines the remote environment settings used with this profile. Within the `remote-environment` element, define an `environment` element for each environment variable to be passed to the server. See [remote-environment](#) for details.

**server-authentication-methods**

This element defines the server authentication methods allowed with this profile. See [server-authentication-methods](#) for details.

**password**

This element can be used to specify a user password that the client will send as a response to password authentication.

The password can be given directly in the `string` attribute, or a path to a file containing the password can be given in the `file` attribute, or a path to a program or a script that outputs the password can be given in the `command` attribute.

When using the `command` option to refer to a shell script, make sure the script also defines the user's shell, and outputs the actual password. Otherwise the executed program fails, because it does not know what shell to use for the shell script. For example, if the password string is defined in a file named `my_password.txt`, and you want to use the bash shell, include these lines in the script:

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```

**Caution**

If the password is given using this option, it is extremely important that the `ssh-broker-config.xml` file, the password file, or the program are not accessible by anyone else than the intended user.

**Note**

Any password given with the command-line options will override this setting.

An example connection profile is shown below:

```
<profile name="rock"
  id="id1"
  host="rock.example.com"
  port="22"
  connect-on-startup="no"
```

```

        user="doct">

        <hostkey file="key_22_rock.pub">
        </hostkey>

        <authentication-methods>
            <auth-publickey />
            <auth-password />
        </authentication-methods>

        <server-authentication-methods>
            <auth-server-publickey policy="strict" />
        </server-authentication-methods>

        <server-banners visible="yes" />

        <forwards>
            <forward type="agent" state="on" />
            <forward type="x11" state="on" />
        </forwards>

        <tunnels>
            <local-tunnel type="tcp"
                listen-port="143"
                dst-host="imap.example.com"
                dst-port="143"
                allow-relay="no" />
        </tunnels>

        <remote-environment>
            <environment name="FOO" value="bar" />
            <environment name="QUX" value="%Ubaz" format="yes" />
            <environment name="ZAPPA" value="%Ubaz" />
        </remote-environment>

    </profile>

```

## The static-tunnels Element

The `static-tunnels` setting is used to configure the behaviour of the automatic tunnels. You can create listeners for local tunnels automatically when the Connection Broker starts up. The actual tunnel is formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.

The `static-tunnels` element can contain any number of `tunnel` elements.

### tunnel

The `tunnel` element specifies a static tunnel. It has the following attributes: `type`, `listen-port`, `listen-address`, `dst-host`, `dst-port`, `allow-relay`, and `profile`.



The `type` attribute defines the type of the tunnel. This can be either `tcp` or `ftp`.

- `tcp` specifies a listener for generic TCP tunneling
- `ftp` specifies a listener for FTP tunneling (also the FTP data channels are tunneled)

The `listen-port` attribute defines the listener port number on the local client.

The `listen-address` attribute can be used to define which network interfaces on the client should be listened. Its value can be an IP address belonging to an interface on the local host. Value `0.0.0.0` listens to all interfaces. The default is `127.0.0.1` (localhost loopback address on the client). Setting any other value requires that `allow-relay="yes"`.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = server host).

The `allow-relay` attribute defines whether connections to the listened port are allowed from outside the client host. The default is `no`.

The `profile` attribute specifies the connection profile id that is used for the tunnel.

```
<static-tunnels>
  <tunnel type="tcp"
    listen-address="127.0.0.1"
    listen-port="9000"
    dst-host="st.example.com"
    dst-port="9000"
    allow-relay="no"
    profile="idl" />
</static-tunnels>
```

## The gui Element

The `gui` element is used to adjust the SSH Tectia terminal GUI settings. The `gui` element takes the following attributes: `hide-tray-icon`, `show-exit-button`, `show-admin`, `enable-connector`, and `show-security-notification`. The last two settings have effect only when transparent TCP tunneling is activated on the system. All of these must have `yes` or `no` as the value.

The `hide-tray-icon` attribute controls whether the SSH Tectia icon is displayed in the system tray. The default is `no` (the tray icon is displayed).

The `show-exit-button` attribute controls whether the **Exit** command is displayed in the shortcut menu of the SSH Tectia icon. The default is `yes`.

The `show-admin` attribute defines whether the **Configuration** command is displayed in the SSH Tectia icon shortcut menu. The default is `yes`. If the button is not displayed, the SSH Tectia Configuration tool can be started by running `ssh-TECTIA-configuration.exe`, located by default in directory "C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Broker".

The `enable-connector` attribute defines whether transparent TCP tunneling is active and capturing application connections for tunneling. The default is `yes`.

On Windows, the `show-security-notification` attribute defines whether the SSH Tectia security notifications are shown upon establishing or closing transparent TCP or FTP tunnels. The default is `yes`.

```
<gui hide-tray-icon="no"
      show-exit-button="yes"
      show-admin="yes"
      enable-connector="yes"
      show-security-notification="yes" />
```

## The `filter-engine` Element

The `filter-engine` element handles the settings related to transparent TCP tunneling that has to be separately selected when installing the SSH Tectia Client.



### Note

The `filter-engine` element is read from the global configuration file, if such a file is available (SSH Tectia Client/ConnectSecure is controlled by SSH Tectia Manager). Only when the global configuration file does not contain the `filter-engine` element, this element is read from the user-specific configuration file.

On Unix, the global configuration is stored as `/etc/ssh2/ssh-broker-config.xml`, and on Windows as `"C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Broker\ssh-broker-config.xml"`.

For configuration examples, see these sample files:

- On Unix: `etc/ssh2/ssh-broker-config-example-capture.xml` and `etc/ssh2/ssh-broker-config-example.xml`
- On Windows: `"<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config-example-capture.xml"` and `"<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config-example.xml"`

The top level element is `filter-engine`. It has two attributes: `ip-generate-start` and `ftp-filter-at-signs` (used with SSH Tectia ConnectSecure, only).

The `ip-generate-start` attribute defines the start address of the pseudo IP address space. Pseudo IPs are generated by the Connection Broker when applications do the DNS query through the SSH connection capture component.

With SSH Tectia ConnectSecure, the `ftp-filter-at-signs` attribute can be used with FTP-SFTP conversion when scripts are used to open a connection directly from the FTP/SFTP client to the SFTP server, bypassing any proxies. This attribute defines that SSH Tectia ConnectSecure uses the FTP user name, FTP server name, and FTP server password specified in the FTP script.

The FTP script is expected to specify the username in format `ftp-user@proxy-user@ftp-server` and the password in format `ftp-password@proxy-password`. The @ sign is used to extract the relevant data from the strings.

The `ftp-filter-at-signs` takes `yes` and `no` as values, `no` is the default.

When `ftp-filter-at-signs="yes"`, SSH Tectia ConnectSecure cuts the username string at the first @ sign to extract the `ftp-user` and at the last @ sign to extract the `ftp-server`, and the rest of the string is ignored. Likewise, the passwords string is cut at the last @ sign and the first part is used as the password on the SFTP server.

## Note

Under the `filter-engine` element there can be any amount of elements `network`, `dns`, `filter`, or `rule`. The order of the elements is important, because the filter engine uses the elements in the order they were specified in the configuration file.

### network

The `network` element specifies a "location" where SSH Tectia Client/ConnectSecure is running. By using the `network` element, you can implement location-awareness for SSH Tectia Client/ConnectSecure. It has four attributes: `id`, `address`, `domain`, and `ip-generate-start`.

The `id` attribute specifies a unique identifier for the `network` element. The `address` attribute specifies the address of the network. It can be missing or empty, in which case it is not used. The `domain` attribute contains the domain name of the computer. It can also be missing or empty, in which case it is not used. The `ip-generate-start` attribute defines the start address of the pseudo IP space. If it is defined here, it overrides the `ip-generate-start` attribute of the `filter-engine` element.

### dns

## Note

The `dns` element exists for backward-compatibility reasons. Currently the `rule` element is used for the same settings.

The `dns` element creates a DNS rule for the filter engine. It has six attributes: `id`, `network-id`, `application`, `host`, `ip-address`, and `pseudo-ip`. For their descriptions, see [rule](#) below.

### filter

## Note

The `filter` element exists for backward-compatibility reasons. Currently the `rule` element is used for the same settings.

The `filter` element specifies an action for a connection. It has the following attributes: `dns-id`, `ports`, `action`, `profile-id`, `destination`, `destination-port`, `fallback-to-plain`.

The `dns-id` attribute is a reference to a `dns` element.

For the descriptions of the other attributes, see [rule](#) below.

## rule

The `rule` element specifies how a filtered connection will be handled. It has the following attributes: `application`, `host`, `ip-address`, `pseudo-ip`, `ports`, `action`, `profile-id`, `destination`, `destination-port`, `username`, `hostname-from-app`, `username-from-app`, `fallback-to-plain`.

The `application` attribute can be used to specify one or more applications to which the rule is applied. This can be a regular expression using the `egrep` syntax. For information on the syntax, see [Appendix D](#).

The `host` attribute specifies a target hostname. It can be a regular expression using the `egrep` syntax.

The `ip-address` attribute specifies the target host IP address. It can be a regular expression using the `egrep` syntax. If both the hostname and the IP address are defined, the `host` attribute takes precedence and the `ip-address` attribute is ignored.

The `pseudo-ip` setting has the following effects when the `ip-address` is left empty and the `host` matches:

- When `pseudo-ip="yes"`, the Connection Broker assigns a pseudo IP address for the target host and SSH Tectia Server resolves the real IP address. The pseudo IP addresses should be used when accessing an internal network from the outside, because name resolution for the machines in the internal network is not available from the outside.
- When `pseudo-ip="no"`, a normal DNS query is made for the target hostname. The default value is `no`.

The `ports` attribute can be a single port or a range. A range is specified with a hyphen between two integers (for example `"21-25"`).



## Note

For FTP-SFTP conversion, always specify the port unambiguously if fallback mode is set. Do not use an asterisk (\*), because it causes problems in passive mode file transfer when connected to a plaintext FTP server.

The `action` attribute specifies the action to be done when a filter matches. Its value can be `DIRECT`, `BLOCK`, `TUNNEL`, `FTP-TUNNEL`, or `FTP-PROXY`.

- `DIRECT` causes the connection to be made directly as plaintext without tunneling or FTP-SFTP conversion.
- `BLOCK` causes the connection to be blocked.

- `FTP-TUNNEL` activates transparent FTP tunneling
- `TUNNEL` activates transparent TCP tunneling
- `FTP-PROXY` causes the FTP-SFTP conversion to start and a connection to be made to the Secure Shell SFTP server.

The `profile-id` attribute can be used to specify the connection profile that defines the connection settings.

If the `profile-id` attribute is left empty and `hostname-from-app="yes"` is specified, the Secure Shell connection is made to the server specified by the client application using default settings. If a `profile-id` is specified and also `hostname-from-app="yes"` is specified, or the referred profile has \* (an asterisk) or empty as the value of the `host` attribute, the Secure Shell connection is made to the server specified by the client application using the profile settings.

The `destination` and `destination-port` attributes can be used to define a static destination address and port number that will be used as the end point of the connection instead of the original address and port given by the application.

The `username` attribute can be used to define the user name used for connecting to the Secure Shell server, or you can define the path from where the Connection Broker should retrieve the user name.

The `hostname-from-app` attribute defines whether the Connection Broker should extract the Secure Shell server's host name from data sent by the application, or use a Secure Shell server defined by the connection profile in `profile-id`. The value is `yes` or `no`, and the default is `no`.

When `hostname-from-app="no"`, the tunnel will be created to the Secure Shell server specified in the connection profile referred in the `profile-id` attribute. Note that with transparent tunneling, the connection from the Secure Shell server to the final destination application will be unsecured and in plaintext. To achieve end-to-end security, the Secure Shell server should reside on the same host as the application.

When `hostname-from-app="yes"`, the tunnel will be created to the destination server specified by the application. This setting can be used with both FTP and TCP tunneling and FTP-SFTP conversion. When using `hostname-from-app="yes"`, it is no longer necessary to create a separate connection profile for each destination host. Note that this requires that a Secure Shell server is installed to each destination server (or that `fallback-to-plain` is enabled to allow direct connections to those servers that do not have Secure Shell installed).

The `username-from-app` attribute defines whether the FTP tunneling or FTP-SFTP conversion extracts the user name from data sent by the FTP application. The value is `yes` or `no`. The default is `no`.

When `username-from-app="yes"`, the user name received from the FTP client application is used. This setting can be used with FTP tunneling and FTP-SFTP conversion. This setting will override any user name settings made in a related connection profile. When `username-from-app="no"`, the user name is taken from the connection profile defined with the `profile-id` attribute.

The `fallback-to-plain` attribute can be used to define whether a direct (unsecured) connection is used if creating the tunnel fails or the connection to the Secure Shell server fails. The default value is `no`. Normally, when the secured connection fails when applying a filter rule, the Connection Broker will return a "host not reachable" error.

### Note

Do not enable the `fallback-to-plain` and `pseudo-ip` options at the same time. If they both are enabled, and a secure connection fails, the application will try a direct connection with the pseudo IP, which will not work.

## The logging Element

The `logging` element changes the logging settings that define the log event severities and logging facilities. The element contains one or more `log-events` elements.

### log-events

This element sets the severity and facility of different logging events. The events have reasonable default values, which are used if no explicit logging settings are made. This setting allows customizing the default values.

For the events, `facility` and `severity` can be set as attributes. The events itself should be listed inside the `log-events` element.

The facility can be `normal`, `daemon`, `user`, `auth`, `local0`, `local1`, `local2`, `local3`, `local4`, `local5`, `local6`, `local7`, or `discard`. Setting the facility to `discard` causes the server to ignore the specified log events.

On Windows, only the `normal` and `discard` facilities are used.

The severity can be `informational`, `notice`, `warning`, `error`, `critical`, `security-success`, or `security-failure`.

Any events that are not specifically defined in the configuration file will use the default values. The defaults can be overridden for all remaining events by giving an empty `log-events` element after all other definitions and by setting a severity value for it.

For a complete list of log events, see [Appendix E](#).

## A.3 Backup of Configuration Files

Before you start upgrades or creating test configurations, make a backup of the Connection Broker configuration files where you have made modifications.

On Unix, the backup copies of the configuration files are made manually. The original user-specific configuration file is located in `$HOME/.ssh2/ssh-broker-config.xml`. Copy it to another location of your own choice. In case you need to return to using the backed up file, copy it back to the original location under the original name.

During SSH Tectia upgrades on Windows, a backup copy is automatically made of the earlier Connection Broker configuration files and stored in the user-specific directory:

```
"%APPDATA%\SSH\backup-<version>-<date>"
```

where

`<version>` is the SSH Tectia release

`<date>` is the date of the upgrade.

On Windows, the user-specific Connection Broker configuration file is by default located in `"%APPDATA%\SSH\ssh-broker-config.xml"`. Each time the Connection Broker configuration file is saved, a backup of the old configuration is stored in `"%APPDATA%\SSH\ssh-broker-config-backup.xml"`. In case you need to return to using the backed up file, copy it back to the original location under the original name.

## A.4 Broker Configuration File Syntax

The DTD of the broker configuration file is shown below:

```
<!-- secsh-broker.dtd -->
<!--
<!-- Copyright (c) 2004-2009 SSH Communications Security, Finland -->
<!-- All rights reserved. -->
<!-- -->
<!-- Document type definition for the Connection Broker XML -->
<!-- configuration files. -->
<!-- -->

<!-- The top-level element -->
<!ELEMENT secsh-broker (general?,default-settings?,profiles?,
                        static-tunnels?,gui?,
                        filter-engine?,logging?)>

<!ATTLIST secsh-broker version CDATA #IMPLIED>

<!-- General element. Only "known-hosts" can appear multiple times. -->
<!ELEMENT general
    (crypto-lib|cert-validation|key-stores|
     strict-host-key-checking|host-key-always-ask|
     accept-unknown-host-keys|known-hosts|
     user-config-directory|file-access-control)*>
```

```

<!-- Cryptographic library. -->
<!ELEMENT crypto-lib      EMPTY>
<!ATTLIST crypto-lib
          mode (fips|standard) "standard">

<!-- PKI settings. -->
<!ELEMENT cert-validation
          (ldap-server*,ocsp-responder*,
           curl-prefetch*,dod-pki?,
           ca-certificate*,key-store*)>

<!ATTLIST cert-validation
          end-point-identity-check (yes|no|YES|NO) "yes"
          default-domain           CDATA #IMPLIED
          http-proxy-url           CDATA #IMPLIED
          socks-server-url         CDATA #IMPLIED>

<!ELEMENT ldap-server      EMPTY>
<!ATTLIST ldap-server
          address             CDATA #REQUIRED
          port                CDATA "389">

<!ELEMENT ocsp-responder  EMPTY>
<!ATTLIST ocsp-responder
          url                 CDATA #REQUIRED
          validity-period     CDATA "0">

<!-- CRL prefetch. -->
<!ELEMENT curl-prefetch  EMPTY>
<!ATTLIST curl-prefetch
          interval            CDATA "3600"
          url                 CDATA #REQUIRED>

<!-- CA certificates. -->
<!ELEMENT ca-certificate (#PCDATA)>
<!ATTLIST ca-certificate
          name                CDATA #REQUIRED
          file                CDATA #IMPLIED
          disable-crls        (yes|no|YES|NO) "no"
          use-expired-crls    CDATA "0" >

<!-- Enable DoD PKI compliancy. -->
<!ELEMENT dod-pki         EMPTY>
<!ATTLIST dod-pki
          enable              (yes|no|YES|NO) "no" >

<!ELEMENT key-stores ((key-store|user-keys|identification)*)>

<!ELEMENT key-store EMPTY>
<!ATTLIST key-store

```



```

        type          CDATA #REQUIRED
        init           CDATA #IMPLIED
        disable-crls   (yes|no|YES|NO) "no"
        use-expired-crls CDATA "0" >

<!-- ELEMENT user-keys EMPTY -->
<!-- ATTLIST user-keys
        directory          CDATA #IMPLIED
        poll-interval      CDATA "10"
        passphrase-timeout CDATA "0"
        passphrase-idle-timeout CDATA "0" -->

<!-- ELEMENT identification EMPTY -->
<!-- ATTLIST identification
        file          CDATA #REQUIRED
        base-path     CDATA #IMPLIED
        passphrase-timeout CDATA "0"
        passphrase-idle-timeout CDATA "0" -->

<!-- Available for backward compatibility reasons -->
<!-- ELEMENT strict-host-key-checking EMPTY -->
<!-- ATTLIST strict-host-key-checking
        enable (yes|no|YES|NO) #REQUIRED -->

<!-- Available for backward compatibility reasons -->
<!-- ELEMENT host-key-always-ask EMPTY -->
<!-- ATTLIST host-key-always-ask
        enable (yes|no|YES|NO) #REQUIRED -->

<!-- Available for backward compatibility reasons -->
<!-- ELEMENT accept-unknown-host-keys EMPTY -->
<!-- ATTLIST accept-unknown-host-keys
        enable (yes|no|YES|NO) #REQUIRED -->

<!-- ELEMENT exclusive-connection EMPTY -->
<!-- ATTLIST exclusive-connection
        enable (yes|no|YES|NO) #REQUIRED -->

<!-- ELEMENT known-hosts (key-store*) -->
<!-- ATTLIST known-hosts
        path          CDATA #IMPLIED
        file          CDATA #IMPLIED
        directory     CDATA #IMPLIED
        filename-format (hash|plain|default) "default" >

<!-- Extended plugin configuration -->
<!-- ELEMENT extended (ext)* -->

<!-- ELEMENT ext (#PCDATA | EMPTY | ext)* -->
<!-- ATTLIST ext

```

```

        name CDATA #REQUIRED>

<!-- Default settings element. No element may appear multiple times.-->
<!ELEMENT default-settings (ciphers, macs,
    transport-distribution, rekey,
    authentication-methods,
    hostbased-default-domain,
    compression, proxy, idle-timeout,
    tcp-connect-timeout, keepalive-interval,
    exclusive-connection, server-banners,
    forwards, extended, remote-environment,
    server-authentication-methods,
    authentication-success-message,
    sftpg3-mode)>

<!-- Server banners. -->
<!ELEMENT server-banners EMPTY>
<!ATTLIST server-banners
    visible (yes|no|YES|NO) "yes">

<!-- Ciphers element. -->
<!ELEMENT ciphers (cipher*)>

<!ELEMENT cipher EMPTY>
<!ATTLIST cipher
    name CDATA #REQUIRED>

<!-- Macs element. -->
<!ELEMENT macs (mac*)>

<!ELEMENT mac EMPTY>
<!ATTLIST mac
    name CDATA #REQUIRED>

<!ELEMENT rekey EMPTY>
<!ATTLIST rekey
    bytes CDATA "0">

<!-- Hostbased default domain. -->
<!ELEMENT hostbased-default-domain EMPTY>
<!ATTLIST hostbased-default-domain
    name CDATA #REQUIRED>

<!-- Authentication methods element. -->
<!ELEMENT authentication-methods (authentication-method|auth-hostbased
    |auth-password|auth-publickey|auth-gssapi
    |auth-keyboard-interactive)*>

<!ELEMENT server-authentication-methods (authentication-method
    |auth-server-publickey
    |auth-server-certificate)*>

```

```

<!ELEMENT auth-server-publickey EMPTY>
<!ATTLIST auth-server-publickey
    policy CDATA #IMPLIED>
    <!-- "strict", "ask", "tofu", -->
    <!-- "advisory" -->

<!ELEMENT auth-server-certificate EMPTY>

<!ELEMENT remote-environment (environment*)>

<!ELEMENT environment EMPTY>
<!ATTLIST environment
    name CDATA #REQUIRED
    value CDATA #REQUIRED
    format (yes|no|YES|NO) "no">

<!-- Transport distribution. -->
<!ELEMENT transport-distribution EMPTY>
<!ATTLIST transport-distribution
    num-transport CDATA #REQUIRED>

<!-- Authentication method. -->
<!ELEMENT authentication-method EMPTY>
<!ATTLIST authentication-method
    name CDATA #REQUIRED>

<!ELEMENT auth-hostbased (local-hostname?)>
<!ELEMENT local-hostname EMPTY>
<!ATTLIST local-hostname
    name CDATA #REQUIRED>

<!ELEMENT auth-password EMPTY>

<!ELEMENT auth-publickey (key-selection?)>
<!ATTLIST key-selection
    policy CDATA #REQUIRED>

<!ELEMENT key-selection (public-key?)>
<!ELEMENT public-key EMPTY>
<!ATTLIST public-key
    type CDATA #REQUIRED>

<!ELEMENT auth-keyboard-interactive EMPTY>

<!ELEMENT auth-gssapi EMPTY>

<!-- User identities. -->
<!ELEMENT user-identities (identity*)>

<!ELEMENT identity EMPTY>

```

```

<!-- identity
      identity-file CDATA #IMPLIED
      file          CDATA #IMPLIED
      hash          CDATA #IMPLIED
      id            CDATA #IMPLIED
      data          CDATA #IMPLIED>

<!-- Password. -->
<!-- ELEMENT password (#PCDATA)>
<!-- ATTLIST password
      string        CDATA #IMPLIED
      file          CDATA #IMPLIED
      command       CDATA #IMPLIED>

<!-- Proxy rules. -->
<!-- ELEMENT proxy EMPTY>
<!-- ATTLIST proxy
      ruleset       CDATA #REQUIRED>

<!-- Idle timeout. -->
<!-- ELEMENT idle-timeout EMPTY>
<!-- ATTLIST idle-timeout
      type (connection) "connection"
      time   CDATA #IMPLIED>

<!-- Connect timeout. -->
<!-- ELEMENT tcp-connect-timeout EMPTY>
<!-- ATTLIST tcp-connect-timeout
      time   CDATA #IMPLIED>

<!-- Keepalive interval. -->
<!-- ELEMENT keepalive-interval EMPTY>
<!-- ATTLIST keepalive-interval
      time   CDATA #IMPLIED>

<!-- Forwards element. -->
<!-- ELEMENT forwards (forward*)>

<!-- ELEMENT forward EMPTY>
<!-- ATTLIST forward
      type (x11|agent) #REQUIRED
      state (on|off|denied) #REQUIRED>

<!-- Compression. -->
<!-- ELEMENT compression EMPTY>
<!-- ATTLIST compression
      name   CDATA #IMPLIED
      level  CDATA #IMPLIED>

<!-- ELEMENT authentication-success-message EMPTY>

```

```

<!-- ATTLIST authentication-success-message
      enable (yes|no|YES|NO) "yes">

<!-- ELEMENT sftpg3-mode EMPTY>
<!-- ATTLIST sftpg3-mode
      compatibility-mode CDATA "tectia">

<!-- ELEMENT user-config-directory EMPTY>
<!-- ATTLIST user-config-directory
      path CDATA "%USER_CONFIG_DIRECTORY%">

<!-- ELEMENT file-access-control EMPTY>
<!-- ATTLIST file-access-control
      enable (yes|no|YES|NO) "no">

<!-- Profiles element. -->
<!-- ELEMENT profiles (profile*)>

<!-- Connection profile. No element may appear multiple times. -->
<!-- ELEMENT profile (hostkey|ciphers|macs,transport-distribution|
      rekey|authentication-methods|user-identities|
      compression|proxy|idle-timeout|
      tcp-connect-timeout|keepalive-interval|
      exclusive-connection|server-banners|
      forwards|tunnels|extended|remote-environment,
      server-authentication-methods|password)>

<!-- ATTLIST profile
      id ID #REQUIRED
      name CDATA #IMPLIED
      host CDATA #REQUIRED
      port CDATA "22"
      protocol CDATA "secsh2"
      connect-on-startup (yes|no|YES|NO) "no"
      user CDATA #IMPLIED
      gateway-profile CDATA #IMPLIED>

<!-- Hostkey. -->
<!-- ELEMENT hostkey (#PCDATA)>
<!-- ATTLIST hostkey
      file CDATA #IMPLIED>

<!-- Tunnels element. -->
<!-- ELEMENT tunnels (local-tunnel*,remote-tunnel*)>

<!-- Local tunnel. -->
<!-- ELEMENT local-tunnel EMPTY>
<!-- ATTLIST local-tunnel
      type CDATA "tcp"
      listen-address CDATA "127.0.0.1"
      listen-port CDATA #REQUIRED

```

```

        dst-host      CDATA "127.0.0.1"
        dst-port      CDATA #REQUIRED
        allow-relay    (yes|no|YES|NO) "no">

<!-- Remote tunnel. -->
<!ELEMENT remote-tunnel EMPTY>
<!ATTLIST remote-tunnel
        type          CDATA "tcp"
        listen-address CDATA "127.0.0.1"
        listen-port    CDATA #REQUIRED
        dst-host       CDATA "127.0.0.1"
        dst-port       CDATA #REQUIRED
        allow-relay    (yes|no|YES|NO) "no">

<!-- Static tunnels element. -->
<!ELEMENT static-tunnels (tunnel*)>

<!-- Static tunnel. -->
<!ELEMENT tunnel EMPTY>
<!ATTLIST tunnel
        type          CDATA "tcp"
        listen-address CDATA "127.0.0.1"
        listen-port    CDATA #REQUIRED
        dst-host       CDATA "127.0.0.1"
        dst-port       CDATA #REQUIRED
        allow-relay    (yes|no|YES|NO) "no"
        profile        CDATA #REQUIRED>

<!-- GUI. -->
<!ELEMENT gui EMPTY>
<!ATTLIST gui
        hide-tray-icon (yes|no|YES|NO) #IMPLIED
        show-exit-button (yes|no|YES|NO) #IMPLIED
        show-admin      (yes|no|YES|NO) #IMPLIED
        enable-connector (yes|no|YES|NO) #IMPLIED
        show-security-notification (yes|no|YES|NO) #IMPLIED>

<!ELEMENT filter-engine (network|dns|filter|rule)*>
<!ATTLIST filter-engine
        ip-generate-start CDATA #IMPLIED
        ftp-filter-at-signs (yes|no|YES|NO) "no">

<!ELEMENT network EMPTY>
<!ATTLIST network
        id          ID #REQUIRED
        address      CDATA #IMPLIED
        domain       CDATA #IMPLIED
        ip-generate-start CDATA #IMPLIED>

<!ELEMENT dns EMPTY>
<!ATTLIST dns

```

```

        id ID #REQUIRED
        network-id IDREF #IMPLIED
        application CDATA #IMPLIED
        host CDATA #IMPLIED
        ip-address CDATA #IMPLIED
        pseudo-ip (yes|no|YES|NO) "no">

<!-- ELEMENT filter EMPTY -->
<!-- ATTLIST filter -->
        dns-id IDREF #REQUIRED
        ports CDATA #REQUIRED
        action (block|direct|tunnel|ftp-tunnel|ftp-proxy|
                BLOCK|DIRECT|TUNNEL|FTP-TUNNEL|FTP-PROXY)
                #REQUIRED
        profile-id CDATA #IMPLIED
        destination CDATA #IMPLIED
        destination-port CDATA #IMPLIED
        fallback-to-plain (yes|no|YES|NO) "no">

<!-- ELEMENT rule EMPTY -->
<!-- ATTLIST rule -->
        application CDATA #IMPLIED
        host CDATA #IMPLIED
        ip-address CDATA #IMPLIED
        pseudo-ip (yes|no|YES|NO) "no"
        ports CDATA #REQUIRED
        action (block|direct|tunnel|ftp-tunnel|ftp-proxy|
                BLOCK|DIRECT|TUNNEL|FTP-TUNNEL|FTP-PROXY)
                #REQUIRED
        profile-id CDATA #IMPLIED
        destination CDATA #IMPLIED
        destination-port CDATA #IMPLIED
        username CDATA #IMPLIED
        hostname-from-app (yes|no|YES|NO) "no"
        username-from-app (yes|no|YES|NO) "no"
        fallback-to-plain (yes|no|YES|NO) "no">

<!-- ELEMENT logging (log-events*) -->

<!-- Log events. -->
<!-- Log event facility. -->
<!-- ENTITY default-log-event-facility "normal" -->

<!-- Log event severity. -->
<!-- ENTITY default-log-event-severity "notice" -->

<!-- ELEMENT log-target EMPTY -->
<!-- ATTLIST log-target -->
        file CDATA #IMPLIED
        type (file|syslog|socket|discard) "file"

```


```

format (syslog|csv|xml) "syslog" >

<!ELEMENT log-events (log-target|#PCDATA)>
<!ATTLIST log-events
    facility (normal|daemon|user|auth|local0|local1|local2
             |local3|local4|local5|local6|local7|discard)
             "&default-log-event-facility;"
    severity (informational|notice|warning|error|critical
             |security-success|security-failure)
             "&default-log-event-severity;">

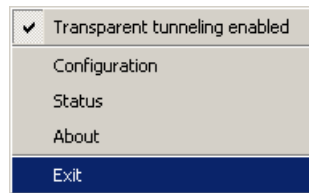
```

## A.5 SSH Tectia Shortcut Menu (Windows)

When SSH Tectia Client (or Connection Broker) is running on Windows, and if so selected in the **General** settings in the SSH Tectia Configuration GUI, the SSH Tectia icon  is displayed on the system tray, typically next to the time display at the bottom of the Windows desktop.

Double-click the SSH Tectia icon to open the Status dialog box, or right-click it to open a shortcut menu.

The contents of the shortcut menu depend on what is defined in the SSH Tectia Configuration GUI in the **General** view (see [Section A.1.2](#)).



**Figure A.43. The shortcut menu of the Status dialog box**

In the menu, you have the following options:

- **Transparent tunneling enabled** is shown in the menu if the transparent TCP tunneling feature has been installed and configured active on the machine when the Connection Broker starts.


You can disable transparent TCP tunneling temporarily by unselecting the option in the shortcut menu. This setting does not affect the Connection Broker configuration, only the current Connection Broker session.

- **Configuration** opens the SSH Tectia Configuration GUI.
- **Status** opens the **SSH Tectia Status** window where you can view information on the Connection Broker. For details, see [Section A.5.1](#).
- **About** shows the installed SSH Tectia Client version.



## A.5.1 SSH Tectia Status Dialog Box (Windows)

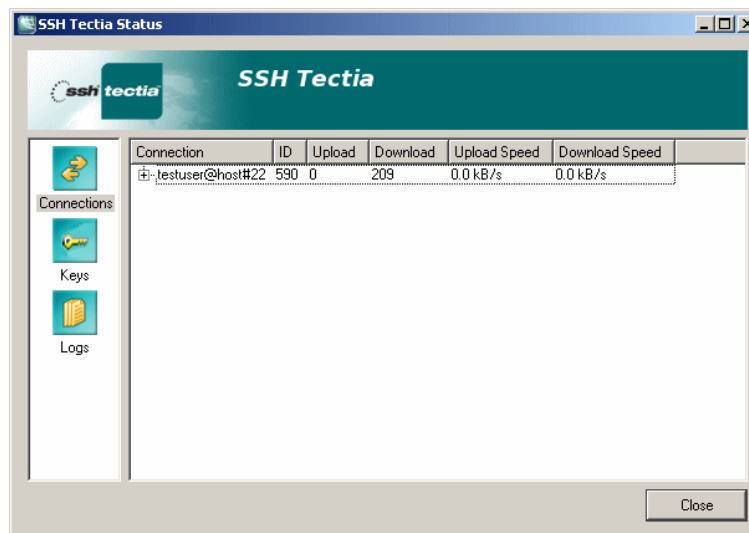
You can view information on the SSH Tectia Connection Broker status in the **SSH Tectia Status** dialog box.

To open the **SSH Tectia Status** dialog box, right-click the SSH Tectia icon  in the system tray or select the **Status** option from the shortcut menu. The left-hand side of the Status dialog box contains links to the different views of the dialog box: the **Connections**, **Keys**, and **Logs** views. Click on the page icons to see the relevant view.

### A.5.1.1 Connections View

The **Connections** view of the **SSH Tectia Status** dialog box displays the currently active secured connections (terminal, tunnel, or SFTP) to and from your computer.

You can close a displayed channel by right-clicking it in this view, and selecting **Terminate channel**.



**Figure A.44. The Connections view of the Status dialog box**

The following information is displayed for each connection:

- **Connection:** The destination of the connection in the *user@host#port* format.
- **ID:** The identifier of the connecting program (a number issued by the Connection Broker).
- **Upload:** Amount of data uploaded (in bytes).
- **Download:** Amount of data downloaded (in bytes).
- **Upload speed:** Upload speed in kilobytes per second.
- **Download speed:** Download speed in kilobytes per second.

### A.5.1.2 Keys View

The **Keys** view of the **SSH Tectia Status** dialog box displays the public keys and certificates that are available for public-key authentication. When you right-click a key name, a short-cut menu appears, and you can select to view detailed information on the key or to clear the key's PIN or passphrase cache, if stored in cache.

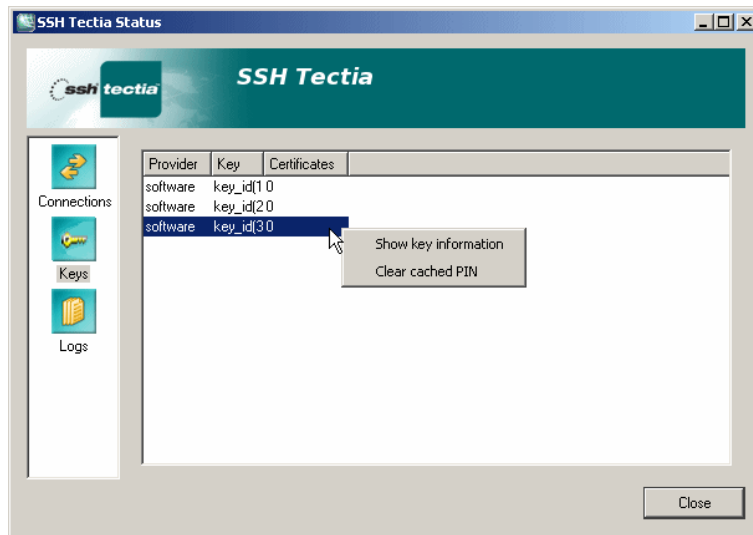


Figure A.45. The Keys view of the Status dialog box

Selecting **Show key information** will display details about the key type and key file location:

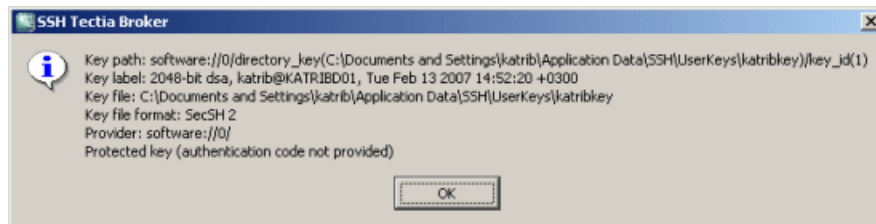
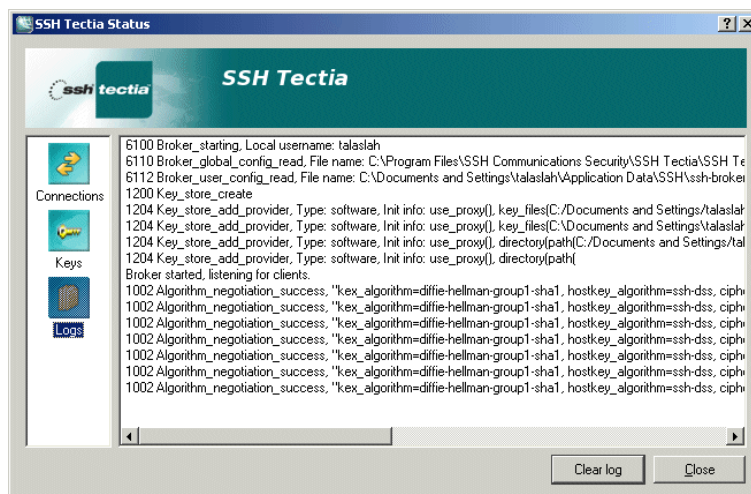


Figure A.46. The key information shown by the Keys view

### A.5.1.3 Logs View

The **Logs** view of the **SSH Tectia Status** dialog box displays logged information of the currently secured connections.




**Figure A.47. The Logs view of the Status dialog box**

You can select text from the log with the mouse and copy it to the clipboard with **Ctrl+C**. Pressing **Ctrl+A** selects all contents of the log. These commands are also available from a shortcut menu (right-click the log).

To clear the log, click **Clear log**.



# Appendix B Configuring SSH Tectia Terminal and File Transfer GUI (Windows)



You can configure the SSH Tectia user interface by clicking the **User Interface Settings** button  on the toolbar, or by selecting the **Edit → User Interface Settings** option.

The different settings categories are visible on the left-hand side of the **Settings** dialog as a tree structure.

Click on a branch to display the settings associated with it. You can change the settings by changing the selections displayed on the right-hand side of the Settings dialog. Note that some of the settings do not take effect until you save the settings and then open a new terminal or file transfer window, or start a new connection.

## Note

SSH Tectia Client includes two configuration tools:

- **SSH Tectia Configuration** tool (behind the  icon) is used to edit the connection settings
- **User Interface Settings** tool (behind the  icon) is used to edit the SSH Tectia terminal and file transfer GUI.

For instructions on the connection configurations, see [Section A.1](#).

## B.1 Defining Global Settings

Global settings are common for all connections to remote host computers. Global settings are saved in the user profile directory ("%APPDATA%\SSH") with the filename `global.dat`.

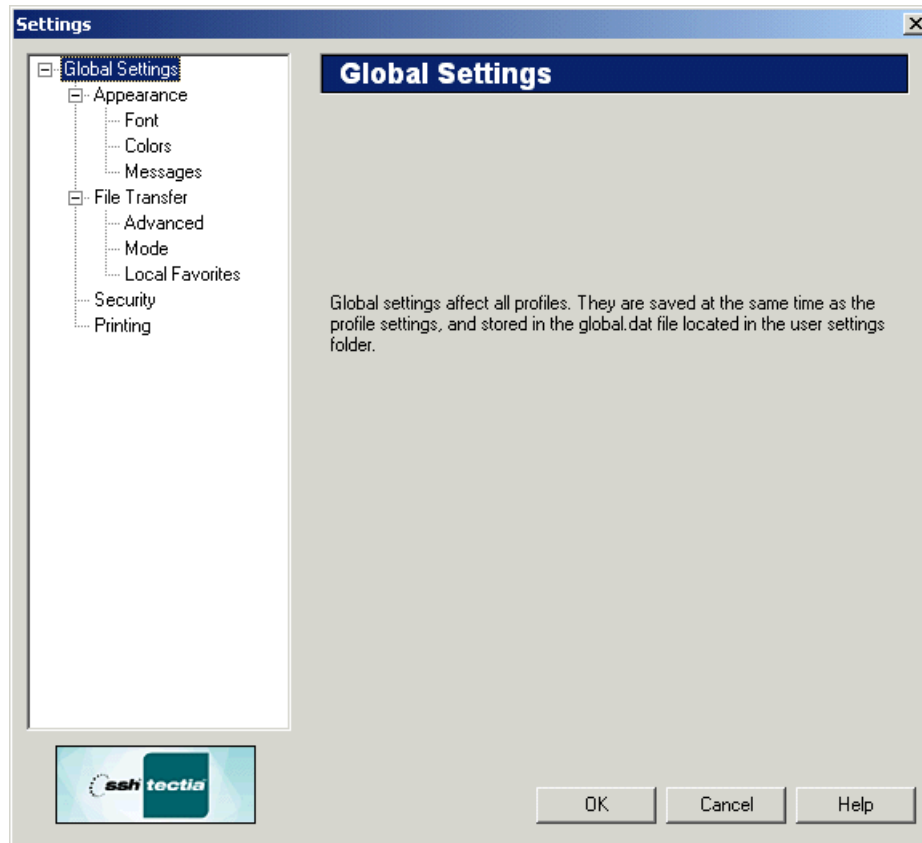
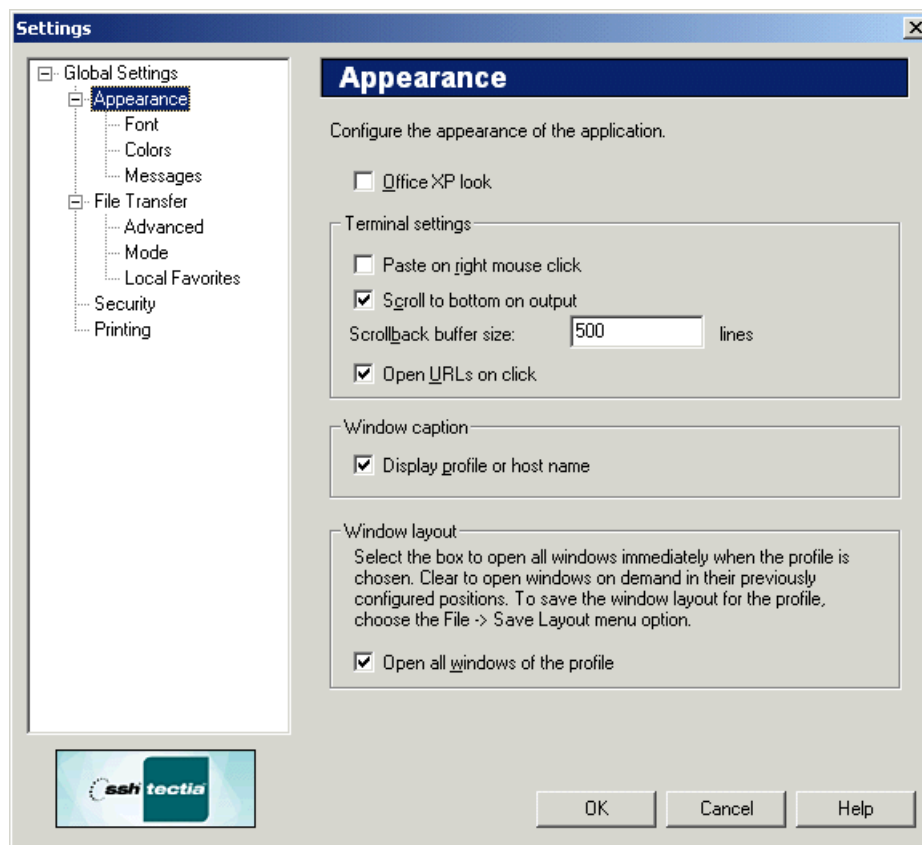


Figure B.1. The Global Settings page of the Settings dialog

### B.1.1 Defining the Appearance

The appearance of the application and the terminal window is configured using the **Appearance** page of the **Settings** dialog.



**Figure B.2. The Appearance page of the Settings dialog**

### Office XP Look

Select the Office XP Look check box to change the way the menu bar and toolbar are displayed to match the visual style of Microsoft Office XP.

### Terminal Settings

With the Terminal Settings options you can define how the terminal window works.

#### Paste on Right Mouse Click

Select this check box to enable fast copying of text on the terminal display. When you have this option selected, you can copy text simply by highlighting it and then paste it by clicking the right mouse button.

#### Scroll to Bottom on Output

Select this check box to have the terminal window scroll to the bottom whenever new text is output. If this option is not selected, you can view the terminal window without the windows scrolling to the bottom every time a new line of text is displayed. By default, this option is on.

### Scrollback Buffer Size

Type in this text box the number of lines that you want to collect into the scrollback buffer. The larger the value, the more you can scroll back the terminal display to view previous terminal output. The default value is 500 lines.

### Open URLs on click

When this check box is selected, links in the terminal window can be opened by clicking them. This option is selected by default.

### Window Caption

The Window Caption settings affect what is displayed in the title bar of the terminal window and the file transfer window.

Select the **Display profile or hostname** check box to have the profile name of the currently connected remote host computer displayed on the title bar if a profile is used. If a profile is not used, the hostname is displayed.

### Window Layout

If you have created a connection profile with several windows open at the same time and saved the layout, all of the windows associated with the profile are normally opened when you select the profile. With the Window Layout option you can override this behavior.

Select the **Open all windows in the profile** check box to open all the windows associated with a profile when the profile is selected. If this option is not selected, the other windows open in their configured positions when you open new windows. By default, this option is on.

## B.1.2 Selecting the Font and Terminal Window Size

The font used in the terminal window can be selected using the **Font** page of the **Settings** dialog. On the same page, you can also select if the size of the terminal window is fixed or adjusted automatically according to the font size.



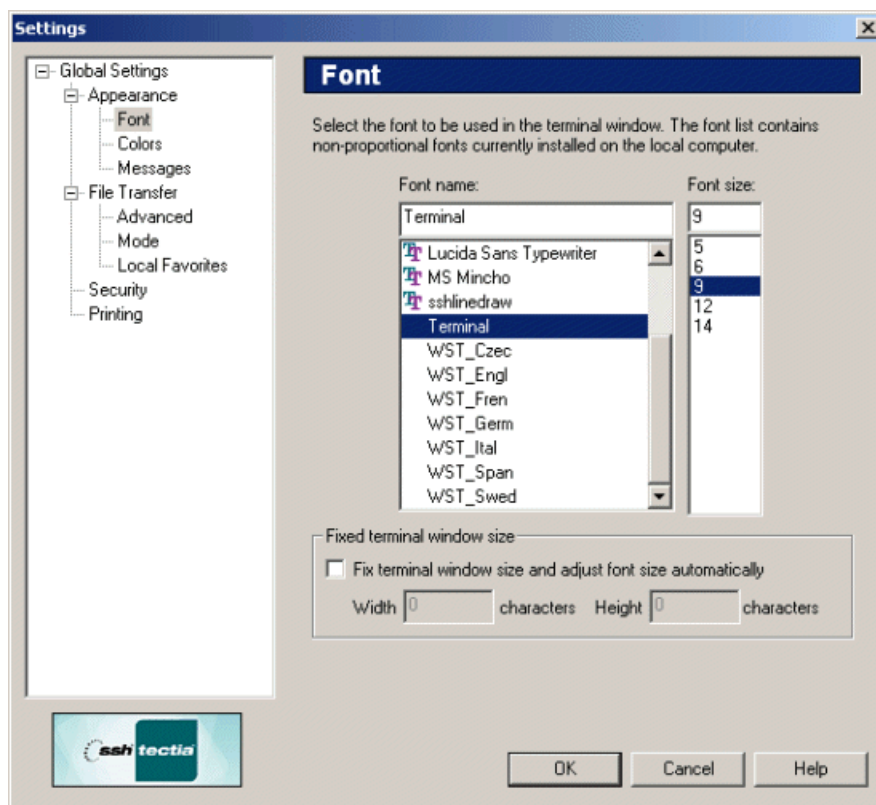


Figure B.3. The Font page of the Settings dialog

## Font

You can select the font to be used and its size. The new font settings affect the terminal window immediately when you click **OK**. To discard the changes, click **Cancel**.

### Font Name

Select the desired font from the **Font Name** list. The list displays the non-proportional (fixed-width) fonts installed in your local computer. Note that proportional fonts are not suitable for the terminal window and therefore are not available for selection.

### Font Size

Select the desired font size from the **Font Size** list. Note that the font size affects the size of the terminal window: the smaller the selected font, the smaller the terminal window. However, after changing the font size, the size of the terminal window can be modified, but the font size remains the selected one.

### Fixed terminal window size

The size of the SSH Tectia Terminal can be defined as fixed. By default, the terminal size is not fixed but adjusted according to the font size. The changes made to the fixed terminal window size setting become effective immediately when you click **OK**. To discard the changes, click **Cancel**.

**Fix terminal window size and adjust font size automatically**

Select this checkbox to use fixed terminal window size. You can define the width and the height of the window. The font size is then adjusted automatically to fit the window, and the font size selection becomes inactive, but you can still select the used font.

**Width**

Define the width of the fixed-size terminal window, in characters. The program suggests 80 characters by default.

**Height**

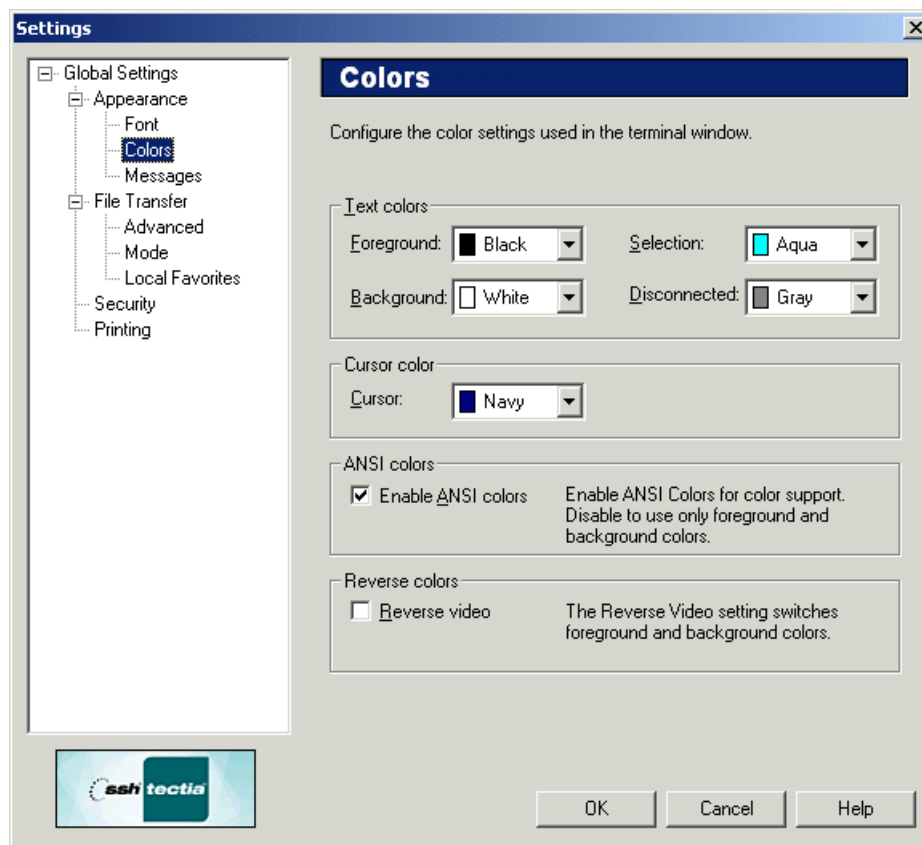
Define the height of the fixed-size terminal window, in characters. The program suggests 24 characters by default.

### **B.1.3 Selecting Colors**

The colors used in the terminal window can be selected using the **Colors** page of the **Settings** dialog. The new color settings are active immediately when you click **OK**.

The color settings defined in Global Settings affect all connection profiles.

Note that changing the terminal colors does not affect what is already visible in the terminal window, but from this point onwards the text output will use the selected color scheme.



**Figure B.4. The Colors page of the Settings dialog**

### Text Colors

The text colors affect the terminal window background color and the color of text in both a connected window and a disconnected window.

#### Foreground

Select the desired foreground color from the drop-down menu. Foreground color is used for text in a window that has a connection to a remote host computer. You can select from sixteen colors. Black is the default foreground color.

#### Background

Select the desired background color from the drop-down menu. You can select from sixteen colors. White is the default background color.

#### Selection

Use the drop-down menu to select the color that is used as the background color when selecting text with the mouse. You can select from sixteen colors. Aqua is the default selection color.

**Disconnected**

Use the drop-down menu to select the color that is used as the foreground color in a terminal window that has no connection to a remote host computer. You can select from sixteen colors. Gray is the default foreground color for a disconnected terminal window.

**Cursor Color**

Select the desired cursor color from the drop-down menu. You can select from sixteen colors. Navy is the default cursor color.

**ANSI Colors**

With ANSI control codes it is possible to change the color of text in a terminal window. With the ANSI Colors setting you can select to use this feature. Even if you disable ANSI colors, you can still select your favorite text and background colors to be used in the terminal window.

Select the **Enable ANSI Colors** check box to allow ANSI colors to be used in the terminal window. By default, ANSI colors are selected.

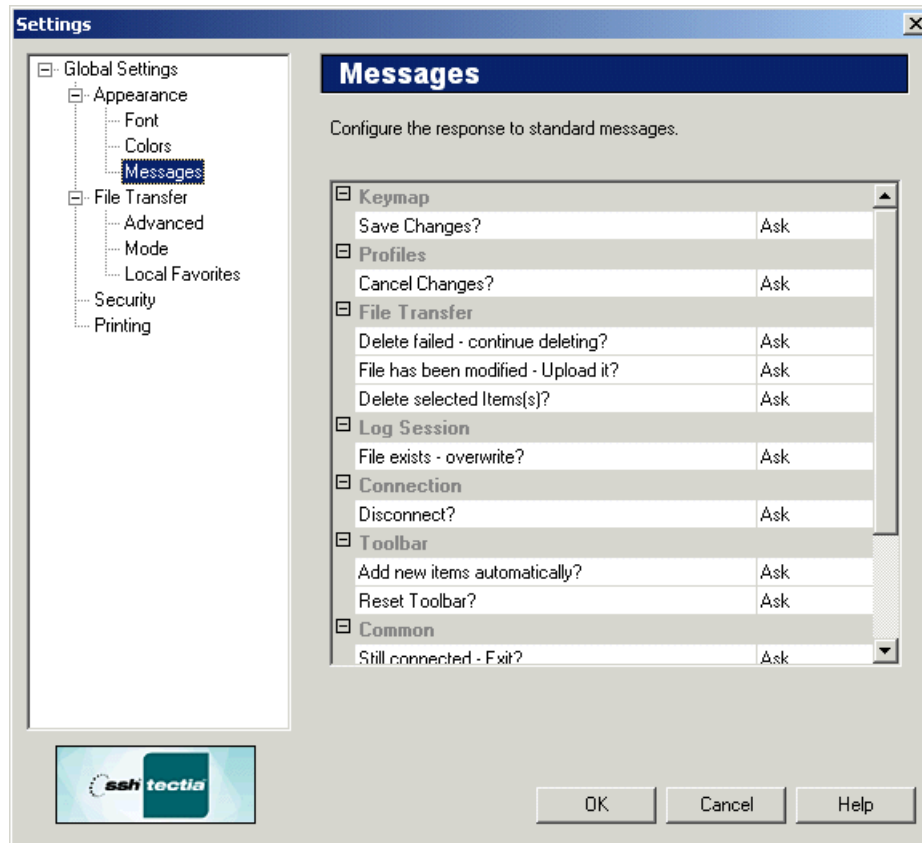
**Reverse Colors**

By reversing the display colors you can quickly change the display from positive (dark on light) to negative (light on dark) to improve visibility.

Select the **Reverse Video** check box to change the foreground color into background color and vice versa. This setting affects the whole terminal window when you click **OK**.

## **B.1.4 Defining Messages**

On the **Messages** page of the **Settings** dialog you can configure default replies to standard messages that normally ask for user confirmation. The messages are listed under several categories.



**Figure B.5. Specifying which confirmation dialogs are displayed**

Each confirmation can be set to automatically accept (**Yes**) or reject (**No**) the action, or to ask the user for confirmation (**Ask**). By default all messages ask the user to confirm the action.

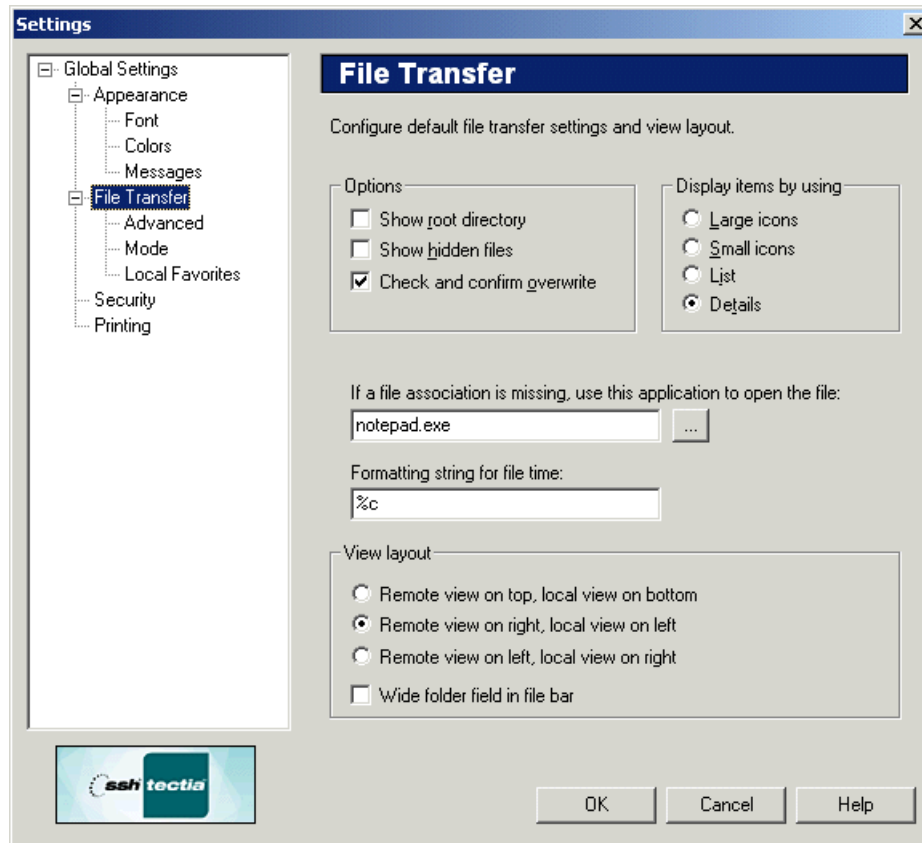
## B.1.5 Defining File Transfer Settings

The default file transfer GUI settings can be configured using the **File Transfer** page of the **User Interface Settings** dialog. The new settings will affect subsequently started file transfer windows.



### Note

The file transfer settings made here only affect the SSH Tectia File Transfer GUI, not the SFTP conversion or command-line tools related to file transfer.



**Figure B.6. The global File Transfer page of the Settings dialog**

### Options

The following options are available:

#### Show Root Directory

Select this check box to show the root directory in the file transfer window by default.

#### Show Hidden Files

Select this check box to show hidden files in the file transfer window by default.

#### Check and Confirm Overwrite

Select this check box if you want the file transfer utility to ask for confirmation when you try to transfer a file that already exists in the target system.

### Display Items by Using

With this setting you can select the default view for the file transfer window by choosing one of the four possible views.

#### Large Icons

Select this option to display the file transfer file view as a Large Icons view. Each file and folder has a large icon associated with it, making for a clear and uncluttered display.

### Small Icons

Select this option to display the file transfer file view as a Small Icons view. Each file and folder has a small icon associated with it. This makes it possible to display several times more items than the Large Icons view.

### List

Select this option to display the file transfer file view as a List view. Each file and folder has a small icon associated with it, and the files and folders are displayed in one column.

### Details

Select this option to display the file transfer folder view as a Details view. The files and folders are displayed with a small icon, their file name, file size, file type, their last modification date and attributes visible.

By clicking on the **Name**, **Size**, **Type**, **Modified** and **Attributes** sort bars located at the top of the File view, you can sort the files and folders based on their file name, file size, file type and the time they were last modified. Clicking the same sort option again reverses the sorting order.

Note that the sort function is not case-sensitive: uppercase text is sorted together with lowercase text.

The file type associations are derived from your local computer. If you have defined a new file type description for files with a certain file name extension, also the files in the remote computer are shown to be of that file type. This makes it easy to recognize particular file types also on the host computer.

### If a file association is missing, use this application to open the file:

SSH Tectia Client uses file type associations in the same way as Windows Explorer does. When you double-click a file in the file transfer window, it is opened using the application with which its file type has been associated.

All file types are not associated with any application. With this field you can define the application to use to open a file that has no file type association. The default application is *Notepad*, which is a reasonable choice for files containing text.

To change the default association for unknown file types, click the button next to the text field. A Select Application dialog is displayed, allowing you to select the desired application.

### Formatting string for file time

In the formatting string field you can type a string that defines how the time and date stamps of the files are displayed in the file transfer window. The default value is `%c`, which means that the date and time will be shown in the format defined in the Windows country settings (locale).

To change the format of the time and date stamps, replace the default value with a string consisting of some of the following character combinations.

---

%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 - 31)
%H	Hour in 24-hour format (00 - 23)
%I	Hour in 12-hour format (01 - 12)
%j	Day of year as decimal number (001 - 366)
%m	Month as decimal number (01 - 12)
%M	Minute as decimal number (00 - 59)
%p	Current locale's A.M. / P.M. indicator for 12-hour clock
%S	Second as decimal number (00 - 59)
%U	Week of year as decimal number, with Sunday as the first day of week (00 - 53)
%w	Weekday as decimal number (0 - 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as the first day of week (00 - 53)



%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 - 99)
%Y	Year with century, as decimal number
%z, %Z	Time-zone name or abbreviation; no characters if time zone is unknown
%%	Percent sign

### View Layout

You can select how the file transfer window positions the local and remote view panes. The following options are available:

- **Remote view on top, local view on bottom**
- **Remote view on right, local view on left**
- **Remote view on left, local view on right**

Select the **Wide folder view in file bar** check box to show fewer buttons in the file bar, leaving more room for the favorite folders lists.

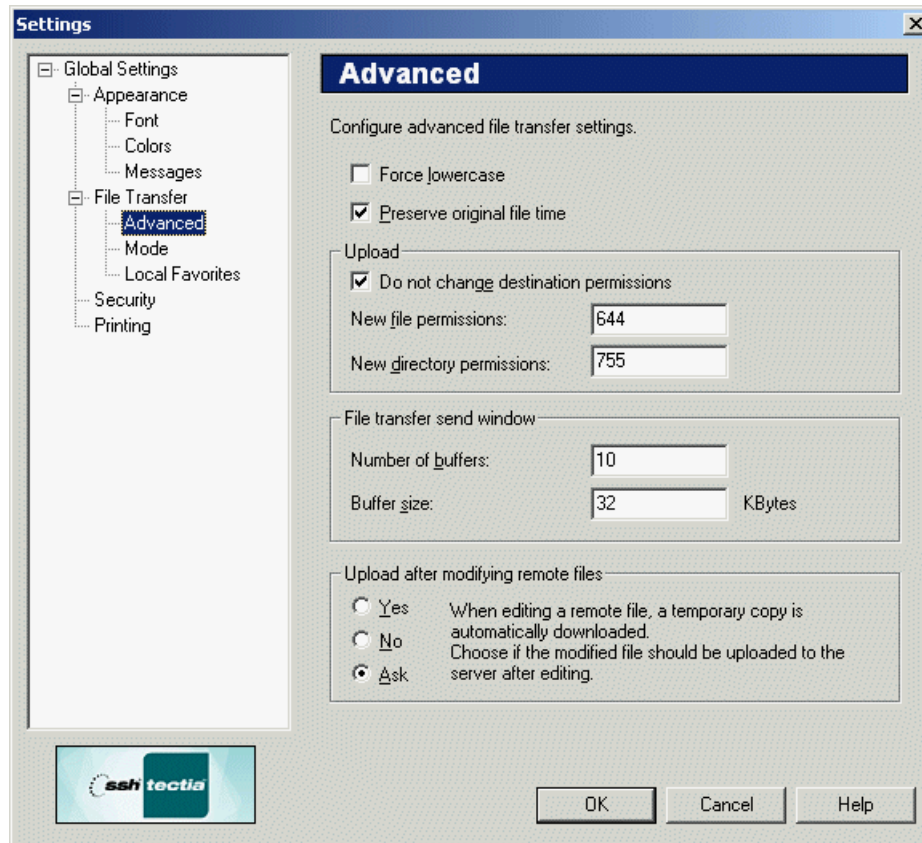
## B.1.6 Defining Advanced File Transfer Options

On the **Advanced** page of the **Settings** dialog you can configure additional file transfer GUI options. The new settings will affect subsequently started file transfer windows.



### Note

The file transfer settings made here only affect the SSH Tectia File Transfer GUI, not the SFTP conversion or command-line tools related to file transfer.



**Figure B.7. The advanced file transfer options**

### Configure advanced file transfer settings.

The following settings affect the file transfer:

#### Force Lowercase

Selecting this option forces lower case file names in file transfers.

#### Preserve Original File Time

Select this check box if you want the transferred files to retain their original time and date stamp values. If this option is not selected, the transferred files will be stamped with the time of the transfer.

### Upload

The following settings affect the upload process:

#### Do not change destination permissions

Select this check box to preserve the file permissions on the server. If the transferred file overwrites an existing file, it will use the same file permissions as the original file. If the file is new, it will use the default permission mask of the server target directory.

Clear this check box to force new file permissions on uploaded files. Define the permissions in **New file permissions** and **New directory permissions** below.

### New file permissions

Type the octal Unix file permission mask (as with the Unix `chmod` command) that is to be used as the value for uploaded files. On Unix systems, the attributes signify the file permissions given to each file:

- **d**: The entry is a directory.
- **r**: The file can be read.
- **w**: The file can be written to.
- **x**: The file can be executed.

On Windows systems, the file may have the following attributes:

- **R**: The file can be read.
- **W**: The file can be written to.
- **X**: The file can be executed (run).

### New directory permissions

Type the octal Unix directory permission mask (as with the Unix `chmod` command) that is to be used as the value for uploaded directories.

On Unix systems, the attribute **d** signifies that the entry is a directory

After the **d** attribute, the **r**, **w** and **x** attributes may be repeated up to three times. If the file does not have a particular attribute, the attribute is replaced with a hyphen (-).

The first three attributes specify the permissions given to the owner of the file, the second triplet specifies the permissions for the user group associated with the file, and the last triplet specifies the permissions given to all other users. For more information on file permissions, see the server documentation.

### File Transfer Send Window

The following settings affect the file transfer process:

#### Number of Buffers

Type the number of buffers used in file transfer. The default value is 10.

#### Buffer size

Type the default buffer size (measured in kilobytes). The default value is 32 kilobytes.

### Upload Locally Modified Remote Files

This selection affects how SSH Tectia Client reacts if you locally edit a file stored in the remote host computer.

**Yes**

If you select this option, the locally modified file is uploaded to the remote host computer.

**No**

If you select this option, the locally modified file is not uploaded to the remote host computer.

**Ask**

If you select this option, SSH Tectia Client asks you to decide if you want to upload a locally modified file.

## B.1.7 Defining File Transfer Mode

The **Mode** page of the **Settings** dialog affects which files are transferred using ASCII mode.



### Note

The file transfer settings made here only affect the SSH Tectia File Transfer GUI, not the SFTP conversion or command-line tools related to file transfer.

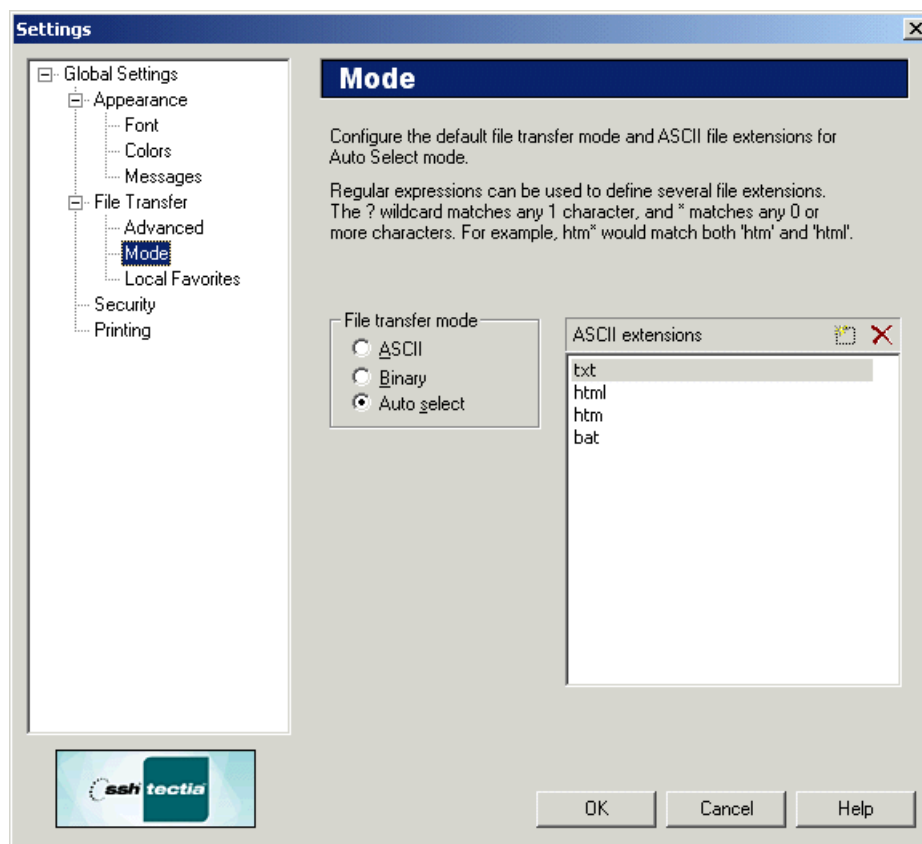


Figure B.8. Selecting the file transfer mode

## File Transfer Mode

Select the default file transfer mode from the following options:

### ASCII

By default all files will be transferred in ASCII mode.

### Binary

By default all files will be transferred in binary mode.

### Auto Select

The files using a file extension specified on the ASCII Extensions list will be transferred in ASCII mode. All other files will be transferred in binary mode.

## ASCII Extensions

Files using a file extension specified in the ASCII Extensions list will be transferred using ASCII mode.

### New

Click the **New** icon (at the top right-hand side of the **ASCII Extensions** list) to add a new file extension to the list. The keyboard shortcut for the New icon is the `Ins` key.

Note that you can use wildcard characters to specify the file extensions. The `?` character matches any 1 character, and the `*` character matches any 0 or more characters. For example `htm*` would match both `htm` and `html`.

### Delete

Select a file extension entry from the list and click the **Delete** icon (at the top right-hand side of the **ASCII Extensions** list) to remove the extension. The keyboard shortcut for the Delete icon is the `Delete` key.

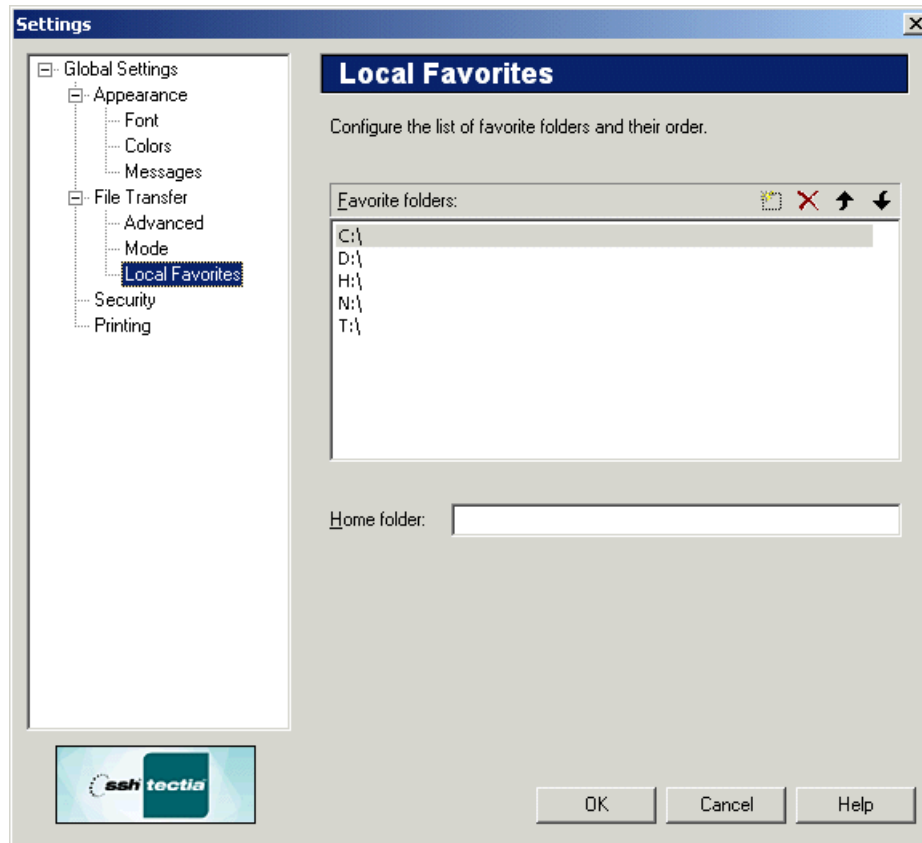
## B.1.8 Defining Local Favorites

On the **Local Favorites** page of the **Settings** dialog you can create a list of commonly used directories on your local computer. These favorites can then be easily selected from a drop-down menu in the File Transfer window.



### Note

The file transfer settings made here only affect the SSH Tectia File Transfer GUI, not the SFTP conversion or command-line tools related to file transfer.



**Figure B.9. Creating a list of most commonly used directories**

### Favorite Folders

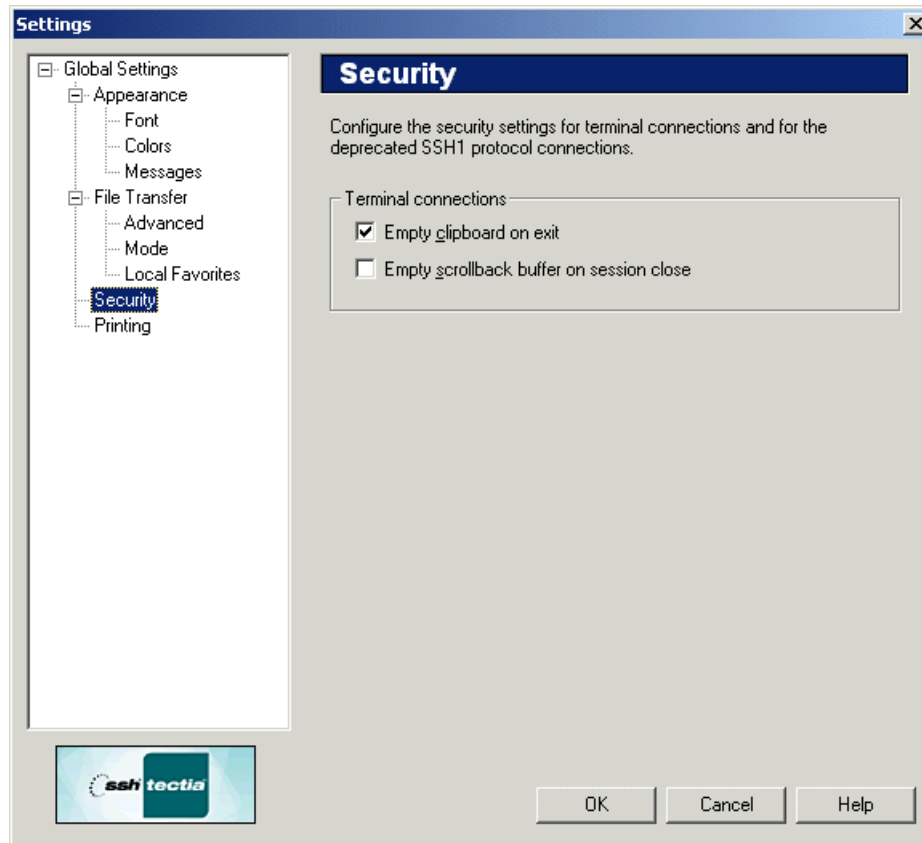
This list contains the favorite folders you have defined for your local computer. Initially the list contains your locally available drives. You can add, remove and sort the favorites by using the **New**, **Delete**, **Up**, and **Down** icons displayed above the list.

### Home Folder

In the **Home Folder** field you can type the directory that is initially displayed in the local view pane of the file transfer window.

## B.1.9 Defining Security Settings

The security settings can be configured using the **Security** page of the Settings dialog.



**Figure B.10. The Security page of the Settings dialog**

### Terminal Connections

The following options are available:

#### Empty Clipboard on Exit

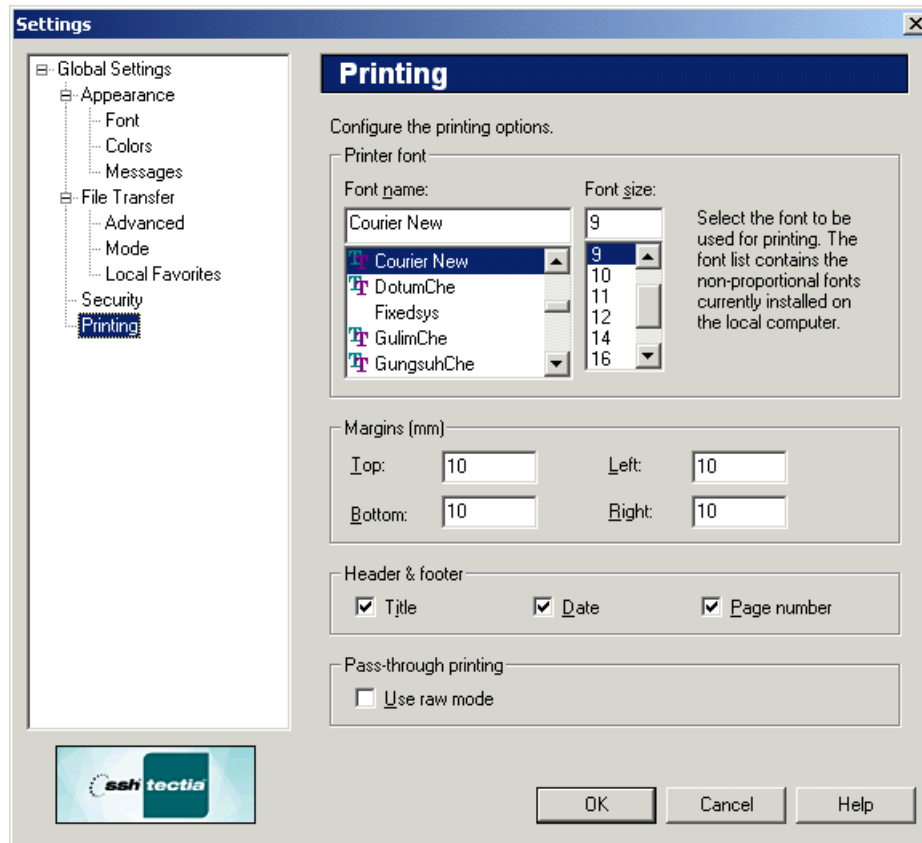
Select this check box to remove anything that was recently copied using the cut and paste operations from the clipboard.

#### Empty Scrollback Buffer on Session Close

Select this check box to empty any remains of the terminal output from the scrollback buffer.

## B.1.10 Printing

The print settings can be configured using the **Printing** page of the Settings dialog.



**Figure B.11. The Printing page of the Settings dialog**

### Printer Font

Select the **Font Name** and **Font Size** to be used in the printed output. Any non-proportional font installed on your system can be selected.

### Margins (mm)

Select the width of the blank border around the page in printed output. The margins for the top, bottom, left and right side of the page can all be specified individually. The default value for all margins is 10 millimeters (or 1 centimeter).

### Header & footer

Select additional information to appear on the printed pages.

**Title** appears at the top left of the page and displays the title of the terminal window (for example `remote-host - SSH Tectia Client`).

**Date** appears at the top right of the page and displays the date and time when the page was printed (for example `15 September 2003, 11:10`). The date and time format is the same as used in Windows.

**Page number** appears at the bottom right of the page (for example `Page 1 of 2`).



### Pass-through printing

Pass-through printing allows the server to print on a client printer using terminal emulation codes.

In raw mode, SSH Tectia Client sends the data to be printed as plaintext to the printer. In this mode, printing for example line graphics does not work.

If not in raw mode, SSH Tectia Client sends the data to be printed to the printer as graphics. This is the default setting and should be used if there are no problems in printing. However, some older printers might not support printing graphics.

### Use raw mode

Select this check box to pass the data to be printed to the printer in raw mode. If you experience printing problems, select or clear this selection as applicable.

## B.2 Using Command-Line Options

For some purposes it may be useful to operate the SSH Tectia Terminal GUI from the command line (command prompt).

The command-line syntax for the SSH Tectia Terminal GUI (`ssh-client-g3.exe`) is as follows:

```
ssh-client-g3 [-r] [-p port] [-u user] [-h host] [profile]
```

The command-line parameters have the following meanings:

`-r`

The `-r` option will reset all customizations made to the user interface (toolbars and menus). A confirmation dialog is displayed.

`-p [port_number]`

The `-p` option specifies the port number used for the connection. If this option is not specified, the port number defined in the default profile is used.

`-u [user_name]`

The `-u` option specifies the user name for the connection. If this option is not specified, the user name defined in the default profile is used.

`-h [host_name]`

The `-h` option specifies the hostname for the connection. If this option is not specified, the hostname defined in the default profile is used.

`[profile]`

If a profile is specified, it must be the last option on the command line. Any command-line parameters override the profile settings. If no profile is specified, the default profile is used.

-f

The -f (or /f) option starts the default SFTP file transfer profile.

For example, the following command would immediately start a connection to a host called `remotehost` and connect as `guest`. The port number is not specified, so the connection would use the port specified in the default profile.

```
ssh-client-g3 -h remotehost -u guest
```

The following command would immediately start a connection to `remotehost` using the settings defined in the profile file `custom.ssh2`.

```
ssh-client-g3 -h remotehost custom.ssh2
```

If the host is not specified (using the -h option) and no profile is specified, the login dialog opens, automatically filled with the values specified on the command line.

For example, the following command would display the login dialog with the port number already defined as 222 and `guest` as the user name.

```
ssh-client-g3 -u guest -p 222
```

A command-line client `sshg3.exe` is also included in SSH Tectia Client for Windows. It can be useful especially for creating scripts. For a description of the `sshg3.exe` syntax, see [sshg3\(1\)](#).

Also several other command-line utilities are included in the SSH Tectia Client installation. For more information, see [Appendix C](#).

## B.3 Customizing the User Interface

This section describes the options for modifying the graphical user interface.

### B.3.1 Saving Settings

When you have made changes to the user interface settings, an asterisk (\*) is displayed on the SSH Tectia GUI title bar, after the name of the current settings file (for example: `default*`). This indicates that the changed settings are not yet permanent - they have not been saved yet.

If you want to make the changes permanent, you can save them for later use. Click the **Save** button on the toolbar, or select the **File** → **Save Settings** to save any changes you have made to your current settings.

The positions of the currently open terminal and file transfer windows can be saved separately with the **File** → **Save Layout** option. If you arrange your window positions and save the layout settings in the default settings file, the windows will be automatically positioned the way you prefer them when you next start the SSH Tectia Terminal.

Note that by default all of the windows will be opened at once. This can be changed on the **Appearance** page of the **Settings** dialog so that the defined windows are opened only when necessary when you open new terminal and file transfer windows. See [Section B.1.1](#).

If you spend a lot of effort specifying the settings, it is a good idea to create backup copies of the modified settings files (`ssh-broker-config.xml`, `global.dat`, and `*.ssh2`) and store them in a safe location. This way you will not have to create your personal settings again if your settings files are lost (for example because of a hardware failure).

### B.3.1.1 Multiple Settings Files

You can save separate settings files for each remote host computer. This can be done by using the **Profiles** option. For more information on using profiles, see [Section A.1.3](#).

## B.3.2 Loading Settings

It is easy to take a previously saved profile into use. Select the **Profiles** option on the Profiles toolbar, or from **File** → **Profiles**, and you will see a menu of previously saved profiles. Click on a profile name, and a connection using the profile settings is started immediately.

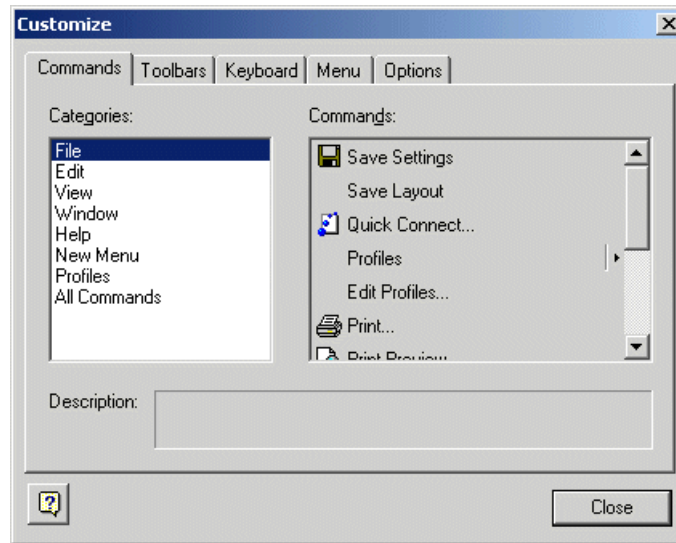
Note that this also works when you are already connected to a remote host computer. Clicking the profile name will start a new, separate connection.

Another way to load the settings for a particular connection is to double-click the settings file name for example in Windows Explorer. When SSH Tectia Client is installed, files with the extension `.ssh2` are associated with the SSH Tectia software. This means that you can start SSH Tectia Client with any settings file loaded by double-clicking the settings file.

If you regularly connect to several remote host computers, you can create shortcuts to the corresponding settings files for example on the Windows desktop. This way you can quickly open the desired connection with the relevant settings already defined, just by clicking on an icon on the desktop.

## B.3.3 Customize Dialog

Select **View** → **Customize** to modify the menu options, toolbars layout, keyboard mapping, menu settings, and general preferences. Note that you can have only one terminal window open when using the **Customize** option.



**Figure B.12.** Use the Customize dialog to modify the user interface settings

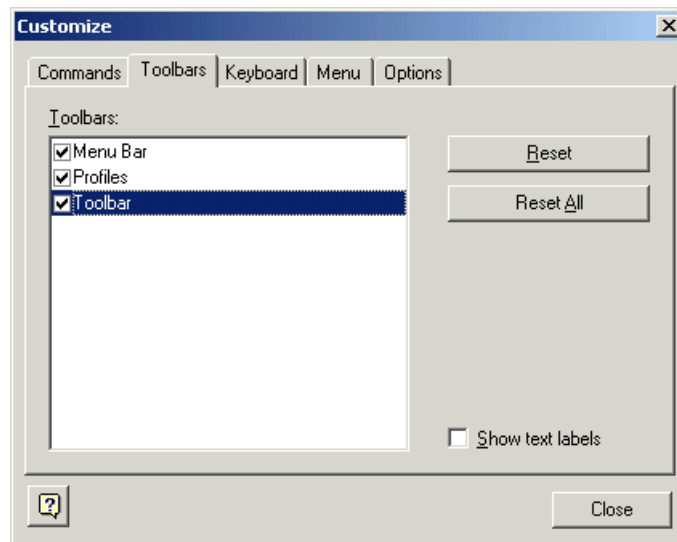
Click on the tabs at the top of the dialog to switch between different pages:

#### **Commands** tab

Select the **Commands** tab to move individual menu options. Select the menu category from the list on the left, and then use the mouse to drag menu options into the menus or toolbars displayed in the SSH Tectia Terminal GUI.

#### **Toolbars** tab

On the **Toolbars** tab, select those toolbars that you want displayed in the SSH Tectia Terminal GUI.

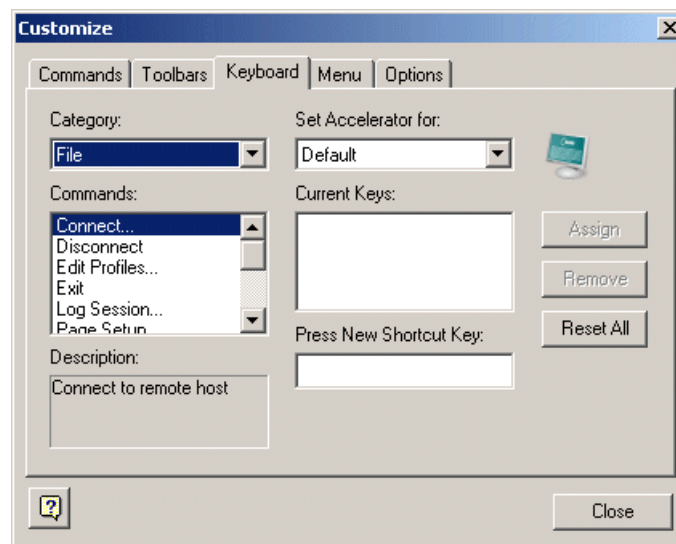


**Figure B.13.** Selecting toolbars

- **Reset:** Select the toolbar that you want to restore to its initial settings and click the **Reset** button to discard the changes you have made.
- **Reset All:** Click the **Reset All** button to discard the changes you have made to all of the toolbars.
- **Show Text Labels:** Select either the **Profiles** or the **Toolbar** option and then select the **Show text labels** check box to display text labels on these toolbars. Text labels clarify the toolbar icons, but also take up space.

### Keyboard tab

Select the **Keyboard** tab to define shortcut keys for the menu commands.



**Figure B.14. Customizing the keyboard**

Use the **Category** menu to select the category of the accelerator key you want to modify. The categories are based on the menu hierarchy.

Use the **Commands** menu to select a specific command from the selected category.

The **Description** box displays a brief description of the currently selected command.

Use the **Set Accelerator for** menu to select the profile that you want to associate with the current keyboard configuration.

The **Current Keys** field shows the currently assigned accelerator keys.

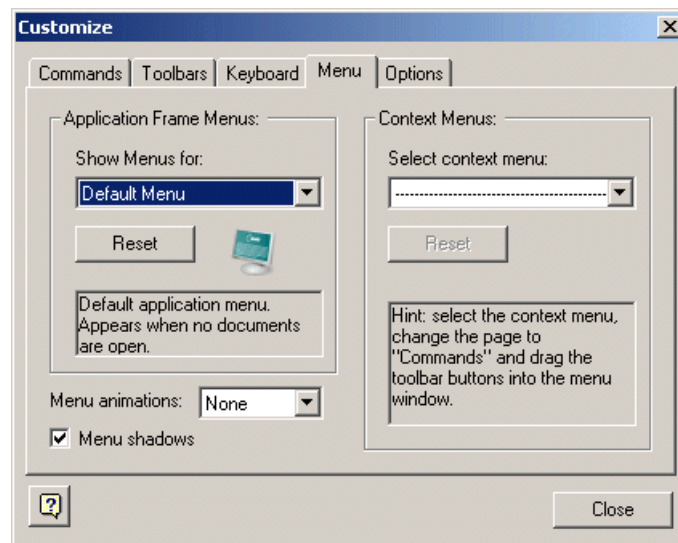
Click on the **Press New Shortcut Key** field to activate it. Then press the combination of keys on the keyboard that you want to associate with the currently selected command.

- **Assign:** Click **Assign** to add the definition from the Press New Shortcut Key field to the Current Keys field.

- **Remove:** Select a key assignment from Current Keys field and click **Remove** to delete it.
- **Reset All:** Click **Reset All** to undo all your changes and reset the keyboard assignments. A confirmation dialog will be displayed.

### Menu tab

Select the **Menu** tab to define the menu settings.



**Figure B.15. Customizing the menus**

- **Application Frame Menus:** Select the menu setup you want to change from the **Show Menus For** drop-down menu. By default, only *Default Menu* is available for editing.

Click **Reset** to reset the menus to their original configuration.

- Use the **Menu Animations** drop-down menu to select the type of menu animations. The available options are *None*, *Unfold*, *Slide*, and *Fade*.
- Select the **Menu Shadows** check box to display shadows under open menus.
- **Context Menus:** Use the **Select Context Menu** drop-down menu to display any of the shortcut (or popup) menus:
  - *File Local Menu 1* is displayed in the local view of the file transfer window when you do not have a file selected.
  - *File Local Menu 2* is displayed in the local view of the file transfer window when you have a file selected.
  - *File Remote Menu 1* is displayed in the remote view of the file transfer window when you do not have a file selected.

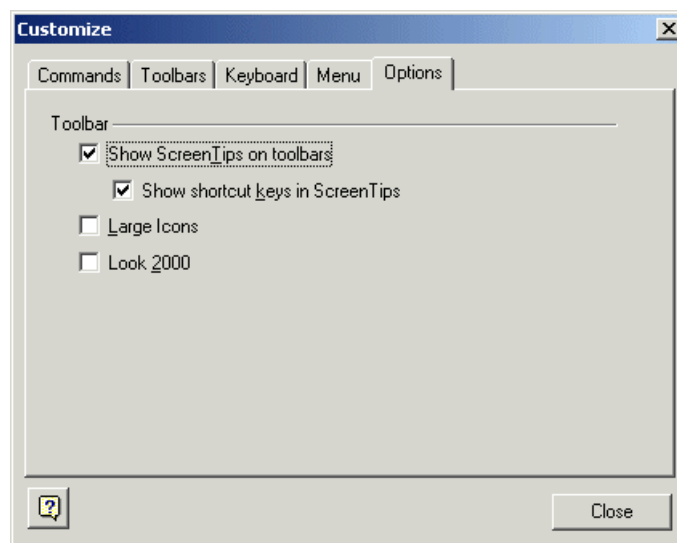
- *File Remote Menu 2* displayed in the remove view of the file transfer window when you have a file selected.
- *Terminal Popup menu* is displayed when you right-click in the terminal window.

You can click the **Commands** tab and drag menu options into the shortcut menus (and remove items from the shortcut menus by dragging them from the menu).

- **Reset:** Click **Reset** to reset the menus to their original configuration.

### Options tab

Select the **Options** tab to change general user-interface options.



**Figure B.16. Customizing general user-interface options**

Select the **Show ScreenTips on Toolbars** check box to display a short help text when you place the mouse pointer over a toolbar button.

Select the **Show Shortcut Keys in ScreenTips** check box to see the possible keyboard shortcut displayed in addition to the short help text.

Select the **Large Icons** check box to display large toolbar icons.

Select the **Look 2000** check box to enable Windows-2000-style features in the user interface. This option affects mainly the style of the toolbar handles.

### Help

Click **Help** to display the online help.

### Close

Click **Close** to stop customizing.

## B.3.4 Customizing Toolbars

SSH Tectia Client has a dynamic user interface where you can select the position of the toolbars.



### Note

The File bar displayed in the file transfer window is dynamically created, and therefore it cannot be customized.

You can use the mouse to grab the toolbars by their handles (located on the left-hand end of each toolbar) and move them around the SSH Tectia Terminal GUI. You can have the toolbars floating freely in the window, or anchor them to the top, bottom or either side of the window.

It is also possible to move the individual menu options. This can be done using the **Commands** page of the Customize dialog (see [Section B.3.3](#)).

Changes become permanent even if you do not save the settings, but you can cancel the changes and reset the view to the default settings.

To undo the changes, select **View** → **Reset Toolbars**. A confirmation dialog opens, asking if you really want to discard the changes. If you select **Yes**, the toolbars and menus will return to their original configuration.



# Appendix C Command-Line Tools and Man Pages

SSH Tectia Client is shipped with several command-line tools. Their functionality is briefly explained in the following appendices.

On Unix, the same information is available on the following manual pages:

- [ssh-broker-g3\(1\)](#): Connection Broker – Generation 3
- [ssh-broker-ctl\(1\)](#): Connection Broker control utility
- [ssh-troubleshoot\(1\)](#): utility for collecting system information for troubleshooting purposes
- [sshg3\(1\)](#): Secure Shell terminal client – Generation 3
- [scp3\(1\)](#): Secure Shell file copy client – Generation 3
- [sftpg3\(1\)](#): Secure Shell file transfer client – Generation 3
- [ssh-translation-table\(1\)](#): Secure Shell file transfer translation table
- [ssh-keygen-g3\(1\)](#): authentication key pair generator
- [ssh-keyfetch\(1\)](#): utility for downloading server host keys
- [ssh-cmpclient-g3\(1\)](#): certificate CMP enrollment client
- [ssh-scepclient-g3\(1\)](#): certificate SCEP enrollment client
- [ssh-certview-g3\(1\)](#): certificate viewer
- [ssh-ekview-g3\(1\)](#): external key viewer

For information on the command-line options of SSH Tectia terminal GUI (`ssh-client-g3.exe`) on Windows, see [Section B.2](#).

For a description of the Connection Broker configuration file options, see [ssh-broker-config\(5\)](#).

## ssh-broker-g3

ssh-broker-g3 -- SSH Connection Broker - Generation 3

### Synopsis

```
ssh-broker-g3 [-f, --config-file=FILE] [-D, --debug=LEVEL] [-l, --debug-log-file=FILE]
[--exit] [--reconfig] [--no-gui] [--start-gui] [-h] [-v]
```

### Description

**ssh-broker-g3** (**ssh-broker-g3.exe** on Windows) is a component of SSH Tectia Client, SSH Tectia Connect-Secure and SSH Tectia MFT Events. It handles all cryptographic operations and authentication-related tasks for SSH Tectia Client and for the client programs **sshg3**, **scpg3**, **sftpg3**, and **ssh-client-g3.exe** (on Windows only).

**ssh-broker-g3** uses the Secure Shell version 2 protocol to communicate with a Secure Shell server.

You can start the Connection Broker manually by using the **ssh-broker-g3** command. This starts **ssh-broker-g3** in the background and all following uses of **sshg3**, **sftpg3**, or **scpg3** will connect via this instance of the Connection Broker instead of starting a new Broker session.

If a command-line client (**sshg3**, **sftpg3**, or **scpg3**) is started when the Connection Broker is not running in the background, the client starts the Broker in *run-on-demand* mode. In this mode, **ssh-broker-g3** will exit after the last client has disconnected.

If there is an **ssh-broker-g3** process running in the run-on-demand mode, and the Connection Broker is started from the command line, the new **ssh-broker-g3** process sends a message to the old **ssh-broker-g3** process to change from the run-on-demand mode to the background mode, keeping the Broker running after the clients disconnect.

### Authentication

The Connection Broker operates automatically as an authentication agent, storing user's public keys and forwarding the authentication over Secure Shell connections. Key pairs can be created with **ssh-keygen-g3**.

On Solaris and AIX, the Connection Broker can also serve OpenSSH clients as an authentication agent.

The public key pairs used for user authentication are by default stored in the `$HOME/.ssh2` directory (`%APPDATA%\SSH\UserKeys` on Windows). See [the section called “Files”](#) for more information.

The Connection Broker automatically maintains and checks a database containing the public host keys used for authenticating Secure Shell servers. When logging in to a server host for the first time, the host's public key is stored in the user's `$HOME/.ssh2/hostkeys` directory (`%APPDATA%\SSH\HostKeys` on Windows). See [the section called “Files”](#) for more information.

## Options

The most important options of **ssh-broker-g3** are the following:

`-f, --config-file=FILE`

Reads the Connection Broker configuration file from *FILE* instead of the default location.

`-D, --debug=LEVEL`

Sets the debug level string to *LEVEL*.

`-l, --debug-log-file=FILE`

Dumps debug messages to *FILE*.

`--exit`

Make the currently running Connection Broker exit. This will terminate all connections.

`--reconfig`

Re-reads the configuration file (`ssh-broker-config.xml`) and takes it into use.

`--no-gui`

On Windows, starts the Connection Broker but does not start the GUI.

This option is used internally when a command-line client is started when the Connection Broker is not running.

`--start-gui`

On Windows, starts the SSH Tectia Configuration GUI if it is not already running.

`-V, --version`

Displays program version and exits.

`-h, --help`

Displays a short summary of command-line options and exits.

On Windows, the help is only shown when running "**ssh-broker-cli.exe -h**" directly from the "C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia AUX\Support Binaries" directory. Normally, **ssh-broker-cli.exe** is never run by the user, but it is automatically called by **ssh-broker-g3.exe**.

## Files

**ssh-broker-g3** uses the following files:

`$HOME/.ssh2/ssh-broker-config.xml`

This is the user-specific configuration file used by **ssh-broker-g3** (and **sshg3**, **scpg3**, and **sftpg3**). The format of this file is described in [ssh-broker-config\(5\)](#). This file does not usually contain any sensitive information, but the recommended permissions are *read/write* for the user, and *not accessible* for others.

On Windows, the user-specific configuration file is located in %APPDATA%\SSH\ssh-broker-config.xml.

\$HOME/.ssh2/random\_seed

This file is used for seeding the random number generator. It contains sensitive data and its permissions should be *read/write* for the user and *not accessible* for others. This file is created the first time the program is run and it is updated automatically. You should never need to read or modify this file.

On Windows, the random seed file is located in %APPDATA%\SSH\random\_seed.

\$HOME/.ssh2/identification

This file contains information on public keys and certificates used for user authentication when contacting remote hosts.

With SSH Tectia Client G3, using the `identification` file is not necessary if all user keys are stored in the default directory and you allow all of them to be used for public-key and/or certificate authentication. If the `identification` file does not exist, the Connection Broker attempts to use each key found in the `$HOME/.ssh2` directory. If the `identification` file exists, the keys listed in it are attempted first.

The identification file contains a list of private key filenames each preceded by the keyword `IdKey` (or `CertKey`). An example file is shown below:

IdKey	mykey
-------	-------

This directs the Connection Broker to use `$HOME/.ssh2/mykey` when attempting login using public-key authentication.

The files are by default assumed to be in the `$HOME/.ssh2` directory, but also a path to the key file can be given. The path can be absolute or relative to the `$HOME/.ssh2` directory. If there is more than one `IdKey`, they are tried in the order that they appear in the identification file.

On Windows, the identification file is located in %APPDATA%\SSH\identification. Key paths in the file can be absolute or relative to the %APPDATA%\SSH directory. The default user key directory is %APPDATA%\SSH\UserKeys and the default user certificate directory is %APPDATA%\SSH\UserCertificates.

\$HOME/.ssh2/hostkeys

This is the user-specific default directory for storing the public keys of server hosts. You are prompted to accept new or changed keys automatically when you connect to a server, unless you have set `strict-host-key-checking` to `yes` in the `ssh-broker-config.xml` file. You should verify the key fingerprint before accepting new or changed keys.

When the host key is received during the first connection to a remote host (or when the host key has changed) and you choose to save the key, its filename is stored by default in hashed format. The hashed host key format is a security feature to make address harvesting on the hosts difficult.

The storage format can be controlled with the `filename-format` attribute of the `known-hosts` element in the `ssh-broker-config.xml` configuration file. The attribute value must be `plain` or `hash` (default).

If you are adding the keys manually, the keys should be named with `key_<port>_<host>.pub` pattern, where `<port>` is the port the Secure Shell server is running on and `<host>` is the hostname you use when connecting to the server (for example, `key_22_alpha.example.com.pub`).

If both hashed and plain-text format keys exist, the hashed format takes precedence.

Note that the identification is different based on the host and port the client is connecting to. For example, the short hostname `alpha` is considered different from the fully qualified domain name `alpha.example.com`. Also a connection with an IP, for example `10.1.54.1`, is considered a different host, as is a connection to the same host but different port, for example `alpha.example.com#222`.

On Windows, the user-specific host key files are located in `%APPDATA%\SSH\HostKeys`.

For more information on host keys, see [Section 5.1](#).

`$HOME/.ssh2/hostkeys/salt`

This is the initialization file for hashed host key names.

On Windows, the salt file is located in `%APPDATA%\SSH\HostKeys\salt`.

`/etc/ssh2/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`

This is the configuration file used by **ssh-broker-g3** (and **sshg3**, **scpg3**, and **sftpg3**) that contains the factory default settings. It is not recommended to edit the file, but you can use it to view the default settings. The format of this file is described in [ssh-broker-config\(5\)](#).

On Windows, the default configuration file is located in "C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml".

`/etc/ssh2/ssh-broker-config.xml`

This is the global (system-wide) configuration file used by **ssh-broker-g3** (and **sshg3**, **scpg3**, and **sftpg3**). The format of this file is described in [ssh-broker-config\(5\)](#).

On Windows, the global configuration file is located in `%APPDATA%\SSH\ssh-broker-config.xml`.

`/etc/ssh2/hostkeys`

If a host key is not found in the user-specific `$HOME/.ssh2/hostkeys` directory, this is the next location to be checked for all users. Host key files are not automatically put here but they have to be updated manually by the system administrator (`root`) or by using SSH Tectia Manager.

If the administrator obtains the host keys by connecting to each host, the keys will be by default in the hashed format. In this case, also the administrator's `$HOME/.ssh2/hostkeys/salt` file has to be copied to the `/etc/ssh2/hostkeys` directory.

On Windows, the system-wide host key files are by default located in:

"C:\Documents and Settings\All Users\Application Data\SSH\HostKeys" on pre-Vista Windows.

"C:\ProgramData\SSH\HostKeys" on Windows Vista.

/etc/ssh2/hostkeys/salt

This is the initialization file for hashed host key names. The file has to be copied here manually by the same administrator that obtains the host keys.

On Windows, the salt file for all users is by default located in:

"C:\Documents and Settings\All Users\Application Data\SSH\HostKeys\salt" on pre-Vista Windows.

"C:\ProgramData\SSH\HostKeys\salt" on Windows Vista.

/etc/ssh/ssh\_known\_hosts

This is the default system-wide file used by OpenSSH clients for storing the public key data of known server hosts. It is supported also by SSH Tectia Client.

If a host key is not found in the user-specific `$HOME/.ssh/known_hosts` file, this is the next location to be checked for all users.

The `ssh_known_hosts` file is never automatically updated by SSH Tectia Client or ConnectSecure, since they store new host keys always in the SSH Tectia user-specific directory `$HOME/.ssh2/hostkeys`.

`$HOME/.ssh/known_hosts`

This is the default user-specific file used by OpenSSH clients for storing the public key data of known server hosts. The `known_hosts` file is supported also by SSH Tectia Client.

The `known_hosts` file contains a hashed or plain-text format entry of each known host key and the port used on the server, in case it is non-standard (other than 22). For more information on the format of the `known_hosts` file, see the OpenSSH `sshd(8)` man page.

The `known_hosts` file is never automatically updated by SSH Tectia Client or ConnectSecure, since they store new host keys always in the SSH Tectia directory `$HOME/.ssh2/hostkeys`.

`$HOME/.ssh2/authorized_keys` (on the server host)

This directory is the default location used by SSH Tectia Server for the user public keys that are authorized for login.

On SSH Tectia Server on Windows, the default directory for user public keys is `%USERPROFILE%\ssh2\authorized_keys`.

`$HOME/.ssh2/authorization` (on the server host)

This is the default file used by earlier versions of SSH Tectia Server (**sshd2**) that lists the user public keys that are authorized for login. The file can be optionally be used with SSH Tectia Server G3 (**ssh-server-g3**) as well.

On SSH Tectia Server on Windows, the authorization file is by default located in `%USERPROFILE%\ssh2\authorization`.

For information on the format of this file, see the `ssh-server-g3(8)` man page.

`$HOME/.ssh/authorized_keys` (on the server host)

This is the default file used by OpenSSH server (**sshd**) that contains the user public keys that are authorized for login.

For information on the format of this file, see the OpenSSH `sshd(8)` man page.

## ssh-broker-ctl

`ssh-broker-ctl` -- SSH Tectia Connection Broker control utility

### Synopsis

`ssh-broker-ctl` *command*  
[options...]

### Description

**ssh-broker-ctl** (**ssh-broker-ctl.exe** on Windows) is a control utility for Connection Broker (**ssh-broker-g3**). It can be used, for example, to view the status of Connection Broker, to reconfigure or stop the Connection Broker, or to load private keys to memory.

### Options

The following general options are available:

`-a, --broker-address ADDRESS`

Defines an address to a separate SSH Tectia Connection Broker process to which a connection is made.

The same effect can be achieved by defining a Connection Broker address with environment variable `SSH_SECSH_BROKER`.

## Note

If you are running **ssh-broker-ctl** using a userID other than that of the **ssh-broker-g3** process owner, the **-a** option must be given so that **ssh-broker-ctl** knows where to connect. In this case, you must also run **ssh-broker-ctl** as a privileged user (root).

For example, when a user *SSHBRKR* owns the **ssh-broker-g3** process:

```
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker status -s
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker status --pid
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker list-connections
```

**-D, --debug *LEVEL***

Defines the debug level.

**-e, --charset=*CS***

Defines the character set to be used in the output. The supported character sets are *utf8*, *iso-8895-1*, *latin1*, *iso-8859-15*, *latin9*, and *ascii*.

**-q, --quiet**

Defines that little or no output is to be displayed, depending on the command.

**-s, --short**

Defines that a shorter, more machine readable, output format is to be used.

**--time-format=*FMT***

Defines the time format to be used in the output. The default depends on the system locale settings.

**-v, --verbose**

Defines that more information, if available, is to be output.

**-V, --version**

Prints the version string.

**-w, --wide**

Defines that the output will not be truncated, even if it means long lines.

**-h, --help**

Displays a context-sensitive help text on command-line options. Help is available also on specific commands. For example, to get help on the *status* command, run:

```
ssh-broker-ctl status --help
```

## Commands

**ssh-broker-ctl** accepts the following commands:



`add-key`

Adds a new private key.

`close-channel channel-id ...`

Closes the defined channel. You can also enter multiple channel-IDs to close several channels.

`close-connection connection-id ...`

Closes the defined connection. You can also enter multiple connection-IDs to close several connections.

`connection-status [--show-channels] [--write-hostkey=FILE] connection-ID`

Displays a detailed connection status for the connection ID (the numeric identifier shown by command **list-connections**).

Options:

`--show-channels`

Displays channel information.

`--write-hostkey=FILE`

Writes the host key (public-key or x509 certificate) to the defined file.

`debug [--append] [--clear] [--log-file=FILE] [--monitor] [debug-level]`

Sets the Connection Broker debug level to the defined level. If no `debug-level` parameter is given here, the current debug level is not changed.

Options:

`--append`

Opens the log file in append mode.

`--clear`

Clears the debug settings. Closes any open log files and sets the debug level to 0.

`--log-file=FILE`

Writes all debug messages to the defined file.

`--monitor`

Monitors the Connection Broker debug output in stderr.

`key-passphrase [--all] [--clear] [--passphrase-file= FILE] [--passphrase-string= passphrase]  
key-id / key-hash`

Prompts the user private key passphrase or PIN code.

Options:

`--all`

Prompts passphrase for all known keys that require it.

`--clear`

Clears cached private key data and possible cached authentication code for the key.

`--passphrase-file=FILE`

Instead of prompting, read the passphrase from the defined file.

`--passphrase-string=passphrase`

Instead of prompting for passphrase, use the passphrase provided on command-line.

`list-channels [-s, --short]`

Displays a list of the currently open connection channels, together with channel type and traffic statistics. Displays also the channel ID which is used by other commands to identify the connection.

Options:

`-s, --short`

Displays a one-line description per channel.

`list-connections [-s, --short] [--show-channels]`

Displays a list of the currently open connections, together with connection parameters and traffic statistics. Displays also the connection ID which is used by other commands to identify the connection.

Options:

`-s, --short`

Displays a one-line description per connection.

`--show-channel`

Displays a short description for each open channel.

`list-keys [-s, --short]`

Displays a list of the user private keys, together with the basic key attributes such as the key type, size, and possible file name or key provider information. Outputs also the fingerprint and the identifier of the key. The identifier is used by other Connection Broker commands to identify the private key.

Options:

`-s, --short`

Displays a one-line description per user private key.

`reload`

Rereads the Connection Broker configuration file.

`stop`

Stops the Connection Broker.

`status [-s, --short] [-q, --quiet] [--pid]`

Without parameters, displays short statistics and a configuration summary for the currently running Connection Broker process.

Options:

`-s, --short`

Displays a one-line output with the Connection Broker PID.

`-q`

Outputs nothing; the exit status is 0 if the Connection Broker connection succeeded, and 1 if the connection failed.

`--pid`

Displays the PID, only.

`view-key [-s, --short] [-v, --verbose] [--clear] [--write-key FILE] key-id`

Displays information on the defined key. If the key has certificates, a short summary of them is also shown.

Options:

`--clear`

Clears cached private key data and cached authentication code for the key.

`-s, --short`

Displays a one-line description per key.

`-v, --verbose`

Displays more detailed information on the key or certificate.

`--write-key=FILE`

Writes the public-key or the certificate to the defined file.

## ssh-troubleshoot

`ssh-troubleshoot -- tool for collecting system information`

### Synopsis

`ssh-troubleshoot [options] [command [command-options] ]`

## Description

**ssh-troubleshoot** (**ssh-troubleshoot.cmd** on Windows) is a tool for collecting information on the operating system (its version, patches, configuration settings, installed software components, and the current environment and state) and on the SSH Tectia installation (installed product components and versions, their state, and the global and user-specific configurations). The collected information will be stored in a `tar` file on Unix or a `log` file on Windows. Send the file to the SSH Tectia Technical Support for analysis to help in troubleshooting situations.

## Options

Enter each option separately, they cannot be combined (not valid: `-Dk`).

The following options are available:

`-k, --keep-going`

Defines that the data collecting is continued as long as possible, even after errors are encountered. Not supported on Windows.

`-o, --output FILENAME`

Defines a non-default output file for storing the collected data. Not supported on Windows.

If *FILENAME* is '-', the collected data is added to the standard output. The default output file is created in a temporary archive directory and stored as `ssh-troubleshoot-data-<hostname>-<timestamp>.tar`. The timestamp is in format: `yyyymmdd-hhmmUTC`.

`-u, --user USERNAME`

Defines another user for the **info** command, the default is the current user. This affects the home directory from which the user-specific SSH Tectia configuration files are fetched. Not supported on Windows.

`-q, --quiet`

Suppresses detailed reporting about the command progress, reports only errors.

`-h, --help`

Displays this help text.

## Commands

**ssh-troubleshoot** accepts the following command:

**info**

Gathers information about the system configuration. The collected data will be stored as a `tar` file on Unix or a `log` file on Windows.

Options:

`--include-private-keys`

Collects everything from the specified user's configuration directories, including the private keys. By default, the private keys nor unrecognized files are not included in the result data. This option is not supported on Windows.

Command syntax on Unix:

```
# ssh-troubleshoot info --include-private-keys
```

Command syntax on Windows:

```
# ssh-troubleshoot.cmd info
```

## sshg3

sshg3 -- Secure Shell terminal client - Generation 3

### Synopsis

```
sshg3 [options...]  
profile | [user@] host [#port]  
[command]
```

### Description

**sshg3** (**sshg3.exe** on Windows) is a program for logging in to a remote machine and executing commands on a remote machine. **sshg3** provides secure, encrypted communication channels between two hosts over an unsecured network. It can be used to replace the unsecured **rlogin**, **rsh**, and **telnet** programs. Also X11 connections and arbitrary TCP/IP ports can be forwarded over secure channels with **sshg3**.

To connect to a remote host using **sshg3**, give either the name of a connection profile defined in the `ssh-broker-config.xml` file (*profile*) or the IP address or DNS name of the remote host, optionally with the remote username and the port of the Secure Shell server (*[user@]host[#port]*). If no username is given, the local username is assumed. If no port is given, the default Secure Shell port 22 is assumed. The remote host must be running a Secure Shell version 2 server.

**sshg3** acts as a Connection Broker client and launches the actual Connection Broker process, **ssh-broker-g3** as a transport (in run-on-demand mode), or uses an already running Connection Broker process. The Connection Broker will ask the user to enter a password or a passphrase if they are needed for authentication. Connection Broker uses the configuration specified in the `ssh-broker-config.xml` file.

When the user's identity has been accepted by the server, the server either executes the given command, or logs in to the machine and gives the user a normal shell. All communication with the remote command or shell will be automatically encrypted.

If no pseudo-tty has been allocated, the session is transparent and can be used to securely transfer binary data.

The session terminates when the command or shell on the remote machine exits and all X11 and TCP/IP connections have been closed. The exit status of the remote program is returned as the exit status of **sshg3**.

## Agent Forwarding (Unix)

**ssh-broker-g3** acts as an authentication agent, and the connection to the agent is automatically forwarded to the remote side unless disabled in the `ssh-broker-config.xml` file or on the **sshg3** command line (with the `-a` option).

## X11 Forwarding

If the user is using X11 (the `DISPLAY` environment variable is set), the connection to the X11 display can be automatically forwarded to the remote side in such a way that any X11 programs started from the shell (or command) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set `DISPLAY`. X11 connection forwarding can be allowed in the `ssh-broker-config.xml` file or on the **sshg3** command line (with the `+x` option). By default, X11 forwarding is disabled.

The `DISPLAY` value set by **sshg3** will point to the server machine, but with a display number greater than zero. This is normal, and happens because **sshg3** creates a "proxy" X server on the server machine for forwarding the connections over the encrypted channel.

**sshg3** will also automatically set up the Xauthority data on the server machine. For this purpose, it will generate a random authentication cookie, store it in the Xauthority data on the server, and verify that any forwarded connections carry this cookie and replace it with the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the plain).

## TCP Port Forwarding

Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either in the `ssh-broker-config.xml` file or on the **sshg3** command line (with the `-L` and `-R` options).

## Options

Command-line options override the settings in the `ssh-broker-config.xml` file if the same option has been configured in both places. The following options are available:

`-a, --no-agent-forwarding`

Disables authentication agent forwarding. This is the default value.

`+a`

Enables authentication agent forwarding.

`-B, --batch-mode`

Uses batch mode. Fails authentication if it requires user interaction on the terminal.

Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

`-C`

Disables compression from the current connection.

`+C`

Enables zlib compression for this particular connection.

`-c, --ciphers=LIST`

Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

Enter `help` as the value to view the currently supported cipher names.

`-D, --debug=LEVEL`

Sets the debug level. *LEVEL* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.



## Note

Option `-D` only applies on Unix. On Windows, instead of this command-line tool, use the Connection Broker debugging options `-D`, `-l`.



## Note

The debug level can be set only when the **sshg3** command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

`-e, --escape-char=CHAR`

Sets escape character (none: disabled, default: `~`).

`-f, --fork-into-background`

Forks into background mode after authentication (Unix only). Use this option with tunnels and remote commands. Implies `-s` (unless a command is specified). When tunnels have been specified, this option makes **sshg3** stay in the background, so that it will wait for connections indefinitely. **sshg3** has to be killed to stop listening.

`-g, --gateway`

Gateways ports, which means that also other hosts may connect to locally forwarded ports. This option has to be specified before the `"-L"` option. Note the logic of `+` and `-` in this option.

`+g`

Does not gateway ports. Listens to tunneling connections originating only from the localhost. This is the default value. Note the logic of `+` and `-` in this option.

`-i FILE`

Defines that private keys defined in the identification file are used for public-key authentication.

`-K, --identity-key-file=FILE`

Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple `-K` options on command line.

`-L, --localfwd [protocol/] [listen-address:] listen-port:dst-host:dst-port`

Forwards a port on the local (client) host to a remote destination host and port.

This allocates a listener port (*listen-port*) on the local client. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the remote server and another connection is made from the server to a specified destination host and port (*dst-host*:*dst-port*). The connection from the server onwards will not be secure, it is a normal TCP connection.

Giving the argument *protocol* enables protocol-specific forwarding. The protocols implemented are `tcp` (default, no special processing), `ftp` (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session), and `socks`.

With the `socks` protocol, the syntax of the argument is `"-L socks/[listen-address:]listen-port".` When this is set, SSH Tectia Client or ConnectSecure will act as a SOCKS server for other applications, creating forwards as requested by the SOCKS transaction. This supports both SOCKS4 and SOCKS5.

If *listen-address* is given, only that interface on the client is listened. If it is omitted, all interfaces are listened.

`-l, --user=USERNAME`

Logs in using this username.

`-m, --macs=LIST`

Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

```
--mac hmac-sha1-96,hmac-md5,hmac-md5-96
```

Enter `help` as the value to view the currently supported MAC names.



`-o option`

Processes an option as if it was read from a SSH Tectia Client 4.x-style configuration file. The supported options are `ForwardX11`, `ForwardAgent`, `AllowedAuthentications` and `PidFile`. For example, `-o "ForwardX11=yes"`. Also `-o "PidFile=/tmp/sshg3.pid"` makes `sshg3` to store its process ID into file `/tmp/sshg3.pid` if it goes into background.

`-P, --password=PASSWORD | file://PASSWORDFILE | extprog://PROGRAM`

Sets user password that the client will send as a response to password authentication. The `PASSWORD` can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (`--password=file://PASSWORDFILE`), or entering a path to a program or script that outputs the password (`--password=extprog://PROGRAM`).

When using the `extprog://` option to refer to a shell script, make sure the script also defines the user's shell, and outputs the actual password. Otherwise the executed program fails, because it does not know what shell to use for the shell script. For example, if the password string is defined in a file named `my_password.txt`, and you want to use the bash shell, include these lines in the script:

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```



## Caution

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

`-p, --port=PORT`

Connects to this port on the remote host. A Secure Shell server must be listening on the same port.

`-q`

Quiet mode, reports only fatal errors.

`-R, --remotefwd [protocol/] [listen-address:] listen-port:dst-host:dst-port`

Forwards a port on the remote (server) host to a destination host and port on the local side.

This allocates a listener port (`listen-port`) on the remote server. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified destination host and port (`dst-host:dst-port`). The connection from the client onwards will not be secure, it is a normal TCP connection.

Giving the argument `protocol` enables protocol-specific forwarding. The protocols implemented are `tcp` (default, no special processing) and `ftp` (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session).

If *listen-address* is given, only that interface on the server is listened. If it is omitted, all interfaces are listened.

`-S, --no-session-channel`

Does not request a session channel. This can be used with port-forwarding requests if a session channel (and tty) is not needed, or the server does not give one.

`+S`

Requests a session channel. This is the default value.

`-s, --subsystem subsystem remote_server`

Sets a subsystem or a service to be invoked on the remote server. The subsystem is specified as a remote command. For example: `sshg3 -s sftp <server>`

`-t, --tty`

Allocates a tty even if a command is given.

`-v, --verbose`

Uses verbose mode. More information or error diagnostics are output if a connection fails.

`-w`

Does not try an empty password.

`+w, --try-empty-password`

Tries an empty password.

`-x, -X, --no-x11-forwarding`

Disables X11 connection forwarding. This is the default value.

`+x, +X`

Enables X11 connection forwarding.

`-z, --broker-log-file=FILE`

Sets the Connection Broker log file to *FILE*. This option works only if **ssh-broker-g3** gets started by this process).

`--abort-on-failing-tunnel`

Aborts if creating a tunnel listener fails (for example, if the port is already reserved).

`--allowed-authentications=METHODS`

Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. Enter `help` as the value to view the currently supported authentication methods.

`--compressions=METHODS`

Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter `help` as the value to view the currently supported compression methods.

`--exclusive`

Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

`--identity ID`

Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

`--identity-key-hash ID`

Defines the private key used in user authentication with the corresponding public key hash.

`--identity-key-id ID`

Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

`--keep-alive=VALUE`

Defines how often keep-alive messages are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

`--remote-environment name=VALUE`

When this option is used, the defined environment variables are passed to the server from the client side. The environment variables are applied on the server when requesting a command, shell or subsystem.

Note that the server can restrict the setting of environment variables.

You can also configure the environment variables to be passed to the server in the `ssh-broker-config.xml` configuration file with the `<remote-environment>` element in the `default-settings` and `per profile`. See [remote-environment](#).

If the same variable is entered on the command-line client and configured in the `ssh-broker-config.xml`, the command-line version will be used.

`--remote-environment-format name=VALUE`

The defined environment variables are passed to the server from the client side. The Connection Broker processes the value before sending it to the server.

You can use `%U` in the value to indicate a user name. The Connection Broker replaces the `%U` with the actual user name before sending it to the server.

For more information, see the `--remote-environment` option above.

`--tcp-connect-timeout=VALUE`

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter the value as a positive number.

`-V, --version`

Displays program version and exits.

`-h, --help`

Displays a short summary of command-line options and exits.

The command can be either of the following ones:

`remote_command [arguments] ...`

Runs the command on a remote host.

`-s service`

Enables a service in remote server.

## Escape Sequences

**sshg3** supports escape sequences to manage a running session. For an escape sequence to take effect, it must be typed directly after a newline character (press **Enter** first). The escape sequences are not displayed on screen during typing.

The following escape sequences are supported:

`~.`

Terminates the connection.

`~Ctrl-Z`

Suspends the session.

`~~`

Sends the escape character literally.

`~#`

Lists forwarded connections.

`~-`

Disables the escape character irrevocably.

`~?`

Displays a summary of escape sequences.

`~r`

Initiates rekeying manually.

`~s`

Gives connection statistics, including server and client version, packets in, packets out, compression, key exchange algorithms, public-key algorithms, and symmetric ciphers.

~C

Gives statistics for individual channels (data window sizes etc). This is for debugging purposes.

~V

Dumps the client version number to stderr (useful for troubleshooting).

## Environment Variables

Upon connection, the Secure Shell server will automatically set a number of environment variables that can be used by **sshg3**. The exact variables set depend on the Secure Shell server. The following variables can be used by **sshg3**:

### DISPLAY

The `DISPLAY` variable indicates the location of the X11 server. It is automatically set by the server to point to a value of the form `hostname:n` where `hostname` indicates the host on which the server and the shell are running, and `n` is an integer greater than or equal to 1. **sshg3** uses this special value to forward X11 connections over the secure channel.

The user should normally not set `DISPLAY` explicitly, as that will render the X11 connection unsecured (and will require the user to manually copy any required authorization cookies).

### HOME

The user's home directory.

### LOGNAME

Synonym for `USER`; set for compatibility with systems using this variable.

### MAIL

The user's mailbox.

### PATH

Set to the default `PATH`, depending on the operating system or, on some systems, `/etc/environment` or `/etc/default/login`.

### SSH SOCKS\_SERVER

The address of the `SOCKS` server used by **sshg3**.

### SSH2\_AUTH\_SOCKET

If this exists, it is used to indicate the path of a Unix-domain socket used to communicate with the authentication agent (or its local representative).

### SSH2\_CLIENT

Identifies the client end of the connection. The variable contains three space-separated values: client IP address, client port number, and server port number.

**SSH2\_ORIGINAL\_COMMAND**

This will be the original command given to **sshg3** if a forced command is run. It can be used, for example, to fetch arguments from the other end. This does not have to be a real command, it can be the name of a file, device, parameters or anything else.

**SSH2\_TTY**

This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

**TZ**

The time-zone variable is set to indicate the present time zone if it was set when the server was started (the server passes the value to new connections).

**USER**

The name of the user.

For a list of variables set by SSH Tectia Server, see the `ssh-server-g3(8)` man page.

## Exit Values

On successful execution, **sshg3** returns normally 0 (zero) as the exit value. If **sshg3** encounters an error, you usually see the reason in an error message. In this case, the exit value is 1.

When executing remote commands, **sshg3** exits with the status of the command run. On successful runs this is normally 0 (zero). The error code 127 is usually returned by the shell if the requested remote command is not found.

## Examples

Connect as the local username to host *remotehost*, port 2222, and open shell:

```
$ sshg3 remotehost#2222
```

Connect to the host specified by the connection profile *profile1* in the `ssh-broker-config.xml` file, and run the `who` command (and exit after running the command):

```
$ sshg3 profile1 who
```

Connect as *user* to host *remotehost*, and open a local port forwarding from port 143 on the client to port 143 on *imapserver*. Do not open shell. Also other hosts may connect to the local port. The connection from *remotehost* to *imapserver* will not be secured:

```
$ sshg3 -L 143:imapserver:143 -g -S user@remotehost
```

## scp3

scp3 -- Secure Shell file copy client - Generation 3

### Synopsis

```
scp3 [options...]  
[ src_profile: | [user@] src_host [#port]: ] src_file...  
[ dst_profile: | [user@] dst_host [#port]: ] dst_file_or_dir
```

### Description

**scp3** (**scp3.exe** on Windows) is used to securely copy files over the network. **scp3** launches **ssh-broker-g3** to provide a secure transport using the Secure Shell version 2 protocol. **ssh-broker-g3** will ask for passwords or passphrases if they are needed for authentication. **scp3** uses the configuration specified in the `ssh-broker-config.xml` file.

Copies between two remote hosts are permitted. The remote host(s) must be running a Secure Shell version 2 server with the **sftp-server** (or **sft-server-g3**) subsystem enabled. SSH Tectia Server has **sft-server-g3** enabled by default.

Any filename may contain a host, user, and port specification to indicate that the file is to be copied to or from that host ( [*user*@] *host* [*#port*] ). If no username is given, the local username is assumed. If no port is given, the default Secure Shell port 22 is assumed. Alternatively, a connection profile defined in the `ssh-broker-config.xml` file (*profile*) can be given.



### Note

When entering a connection profile in the **scp3** command, note that SSH Tectia Client deduces the meaning of the argument differently depending on its format. If there is an @ sign in the given attribute value, SSH Tectia Client always interprets it to be `<username@hostname>`, i.e. not a profile.

Also, if there are dots in a profile name (for example `host.x.example.com`, the dots need to be escaped on command line. On Unix, enter `host\.x\.example\.com`, instead. On Windows, enter `host~.x~.example~.com`, instead. Otherwise the profile name is taken as a host name and the current Windows user name is used for logging in.

The *host* parameter can optionally be enclosed in square brackets ([]) to allow the use of semicolons. The *file* argument can contain simple wildcards: asterisk (\*) for any number of any characters, and question mark (?) for any one character.

For information on special characters in file names, see [the section called “Filename Support”](#).

## Options

The following command-line parameters can be used to further specify the **scp3** options.

**-a** [*arg*]

Transfers files using the ASCII mode, that is, newlines will be converted on the fly. For transfers between SSH Tectia on z/OS and other hosts, this also enables automatic ASCII-EBCDIC conversions. See the `ascii` command in [the section called “Commands”](#).

If the server does not advertise the newline convention, you can give it a hint by giving an argument after **-a**. The default is to set the destination newline convention, but you can specify either one by prefixing the argument with `src:` or `dest:` for source or destination convention, respectively. The available conventions are `dos`, `unix`, and `mac`, using `\r\n`, `\n`, and `\r` as newlines, respectively. An example is shown below:

```
$ scp3 -asrc:unix -adest:dos src_host:src_file dest_host:dest_file
```

**-B**, **--batch-mode**

Uses batch mode. Fails authentication if it requires user interaction on the terminal.

Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

**-b** *buffer\_size\_bytes*

Defines the maximum buffer size for one read/write request (default: 32768 bytes).

**-C**

Disables compression from the current connection.

**+C**

Enables zlib compression for this particular connection.

**-c**, **--ciphers=LIST**

Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

Enter `help` as the value to view the currently supported cipher names.

**-D**, **--debug=LEVEL**

Sets the debug level. *LEVEL* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.



## Note

Option `-D` only applies on Unix. On Windows, instead of this command-line tool, use the Connection Broker debugging options `-D`, `-l`.

## Note

The debug level can be set only when the **scpg3** command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

`-d`

Forces target to be a directory.

`-I`, `--interactive`

Prompts whether to overwrite an existing destination file (does not work with `-B`).

`-i` *FILE*

Defines that private keys defined in the identification file are used for public-key authentication.

`-K`, `--identity-key-file=FILE`

Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple `-K` options on command line.

`-m` *fileperm*[:*dirperm*]

This option can be used only on Unix. Sets the default file and directory permission bits for file upload to Unix servers.

`-N` *max\_requests*

Defines the maximum number of read/write requests sent in parallel (default: 10).

`-O`, `--offset=r<offset> | w<offset> | l<length> | t<length>`

Sets offset. Offset *r<offset>* specifies the start offset in the source file. Offset *w<offset>* specifies the start offset in the destination file. Length *l<length>* specifies the amount of data to be copied. Truncate length *t<length>*, if given, specifies the length to which the destination file is truncated or expanded after the file data has been copied.

`-P`

Preserves the file permissions and the timestamps when both the source and the destination are on Unix filesystems (including z/OS USS). Preserves the timestamps but not the file permissions, if either one, the source or the destination is on Windows. If the destination is on z/OS MVS, none will be preserved.

- `-P port`  
Connects to this Secure Shell port on the remote machine (default: 22).
- `-Q`  
Does not show progress indicator.
- `-r`  
Recurse subdirectories.
- `-u, --unlink-source`  
Removes source files after copying (file move).
- `-v, --verbose`  
Uses verbose mode (equal to `-D 2`).
- `-W, --whole-file`  
Does not try incremental checks. By default (if this option is not given), incremental checks are tried. This option can only be used together with the `--checksum` option.
- `+w, --try-empty-password`  
Tries an empty password.
- `--allowed-authentications=METHODS`  
Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. Enter `help` as the value to view the currently supported authentication methods.
- `--append`  
Appends data to the end of the destination file.
- `--binary`  
Uses binary transfer mode. If the server is SSH Tectia Server for IBM z/OS, the server is requested not to perform ASCII to EBCDIC conversion, and the file is transferred using the Stream format. You can use the `--src-site` and `--dst-site` options to change the values.
- `--checkpoint=b <bytes>`  
Byte interval between checkpoint updates (default: 10 MB). This option can only be used when `--checksum=checkpoint`.
- `--checkpoint=s <seconds>`  
Time interval between checkpoint updates (default: 10 seconds). This option can only be used when `--checksum=checkpoint`.
- `--checksum [=yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint]`  
Uses MD5 or SHA-1 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than `buffer_size_bytes` are not checked. Use `md5-force` or `sha1-force` with small files (default: yes, i.e. use SHA1 checksums in FIPS mode, MD5 checksums otherwise). Use checkpointing when transferring large files one by one.

`--compressions=METHODS`

Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter `help` as the value to view the currently supported compression methods.

`--dst-site=PARAM`

Uses the specified site parameters with the destination files. See the **site** command in [the section called “Commands”](#).

`--exclusive`

Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

`--fips`

Performs the checksums using the FIPS cryptographic library.

`--force-lower-case`

Destination filename will be converted to lowercase characters.

`--identity=ID`

Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

`--identity-key-hash=ID`

Defines the private key used in user authentication with the corresponding public key hash.

`--identity-key-id=ID`

Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

`--keep-alive=VALUE`

Defines how often keep-alive messages (non-operation packages) are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

`--macs=LIST`

Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

```
--mac hmac-sha1-96,hmac-md5,hmac-md5-96
```

Enter `help` as the value to view the currently supported MAC names.

`--overwrite [ =yes | no ]`

Selects whether to overwrite existing destination file(s) (default: `yes`).

`--password= PASSWORD | file://PASSWORDFILE | extprog://PROGRAM`

Sets user password that the client will send as a response to password authentication. The `PASSWORD` can be given directly as an argument to this option (not recommended). Better alternatives are entering a path

to a file containing the password (`--password=file://PASSWORDFILE`), or entering a path to a program or script that outputs the password (`--password=extprog://PROGRAM`).

When using the `extprog://` option to refer to a shell script, make sure the script also defines the user's shell, and outputs the actual password. Otherwise the executed program fails, because it does not know what shell to use for the shell script. For example, if the password string is defined in a file named `my_password.txt`, and you want to use the bash shell, include these lines in the script:

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```



## Caution

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

`--plugin-path=PATH`

Sets plugin path to *PATH*. This is only used in the FIPS mode.

`--prefix=PREFIX`

Adds a prefix to a filename during the file transfer. The prefix is removed after the file has been successfully transferred.

`--src-site=PARAM`

Uses the specified site parameters with the source files. See the **site** command in [the section called “Commands”](#).

`--statistics [ =no | yes | simple | bytes ]`

Chooses the style of the statistics to be shown after a file transfer operation (default: `no`). The options mean:

`no` - no statistics will be created.

`yes` - detailed statistics will be created. You can configure the contents with the `statistics-format` option. The default statistics contents are:

```
"Source: %c:g\n"
"Source parameters: %e\n"
"Destination: %C:G\n"
"Destination parameters: %E\n"
"File size: %s bytes\n"
"Transferred: %t bytes\n"
"Rate: %RB/s\n"
"Start:%x(%Y-%m-%d %H:%M:%S)\n"
```

```
"Stop: %X(%Y-%m-%d %H:%M:%S)\n"
"Time: %y\n"
```

simple - simple one-line statistics will be created. You can configure the contents with the `statistics-format` option. The default statistics contents are:

```
"Transferred %t bytes, file: '%f' -> '%F'\r\n"
```

bytes - basic statistics reporting the transferred bytes will be created. You can configure the contents with the `statistics-format` option. The default statistics contents are:

```
"Transferred %t bytes, file: '%f' -> '%F'\r\n"
```

`--statistics-format=FORMAT STRING`

Chooses the format and the contents of the statistics. Use this option when `--statistics=yes|simple|bytes`. Select the contents for the statistics using the following definitions:

```
%c - source connection: user@host#port or profile
%g - /path/to/source/file
%f - source file name
%e - source parameters (file transfer and dataset parameters)
%C - destination connection: user@host#port or profile
%G - /path/to/destination/file
%F - destination file name
%E - destination parameters (file transfer and dataset parameters)
%s - file size in bytes
%S - file size as "XXyB" (B, kiB, MiB or GiB)
%t - transfer size in bytes
%T - transfer size as "XXyB" (B, kiB, MiB or GiB)
%p - transfer percentage
%q - transfer rate in bit/s
%Q - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r - transfer rate in bytes/s
%R - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%D - current date in format: Wed Jan 28 2009 17:11:52 +0200
%D* - current date
%x - start date in format: Wed Jan 28 2009 17:11:52 +0200
%x* - start date
%X - end date in format: Wed Jan 28 2009 17:11:52 +0200
%X* - end date
%y - elapsed time
%Y - time remaining
%z - ETA or, if the transfer has finished, TOC
%Z - string "ETA" or, if the transfer has finished, "TOC"
%Z(eta)(toc) - string "eta" or, if the transfer has finished, "toc"
```

Where \* indicates the format string used in the `strftime(3)` POSIX function. Note that the format string varies slightly between operating systems. On Linux, the format is defined as follows:

```
%a The abbreviated weekday name according to the current locale.
%A The full weekday name according to the current locale.
%b The abbreviated month name according to the current locale.
```

```

%B The full month name according to the current locale.
%c The preferred date and time representation for the current locale.
%C The century number (year/100) as a 2-digit integer.
%d The day of the month as a decimal number (range 01 to 31).
%D Equivalent to %m/%d/%y.
%e Like %d, the day of the month as a decimal number, but a leading zero is
  replaced by a space.
%E Modifier: use alternative format, see below.
%F Equivalent to %Y-%m-%d (the ISO 8601 date format).
%G The ISO 8601 year with century as a decimal number. The 4-digit year
  corresponding to the ISO week number (see %V). This has the same format
  and value as %y, except that if the ISO week number belongs to the
  previous or next year, that year is used instead.
%g Like %G, but without century, that is, with a 2-digit year (00-99).
%h Equivalent to %b.
%H The hour as a decimal number using a 24-hour clock (range 00-23).
%I The hour as a decimal number using a 12-hour clock (range 01-12).
%j The day of the year as a decimal number (range 001-366).
%k The hour (24-hour clock) as a decimal number (range 0-23);
  single digits are preceded by a blank. (See also %H.)
%l The hour (12-hour clock) as a decimal number (range 1-12);
  single digits are preceded by a blank. (See also %I.)
%m The month as a decimal number (range 01-12).
%M The minute as a decimal number (range 00-59).
%n A newline character.
%O Modifier: use alternative format, see below.
%p Either 'AM' or 'PM' according to the given time value, or the
  corresponding strings for the current locale. Noon is treated as 'pm'
  and midnight as 'am'.
%P Like %p but in lowercase: 'am' or 'pm' or a corresponding string for the
  current locale.
%r The time in a.m. or p.m. notation. In the POSIX locale this is equivalent
  to %I:%M:%S%p.
%R The time in 24-hour notation (%H:%M).
  For a version including the seconds, see %T below.
%s The number of seconds since the Epoch, that is, since 1970-01-01
  00:00:00 UTC.
%S The second as a decimal number (range 00-60). (The range is up to 60 to
  allow for occasional leap seconds.)
%t A tab character.
%T The time in 24-hour notation (%H:%M:%S).
%u The day of the week as a decimal, range 1 to 7, Monday being 1. See
  also %w.
%U The week number of the current year as a decimal number, range 00-53,
  starting with the first Sunday as the first day of week 01.
  See also %V and %W.
%V The ISO 8601:1988 week number of the current year as a decimal number,
  range 01-53, where week 1 is the first week that has at least 4 days in
  the current year, and with Monday as the first day of the week.
  See also %U and %W.
%w The day of the week as a decimal, range 0-6, Sunday being 0. See also %u.

```

```
%W The week number of the current year as a decimal number, range 00-53,
    starting with the first Monday as the first day of week 01.
%x The preferred date representation for the current locale without the time.
%X The preferred time representation for the current locale without the date.
%y The year as a decimal number without a century (range 00-99).
%Y The year as a decimal number including the century.
%z The time-zone as hour offset from GMT. Required to emit RFC 822
    conformant dates (using "%a, %d %b %Y %H:%M:%S %z").
%Z The time zone or name or abbreviation.
%+ The date and time in date(1) format.
    (Not supported in glibc2.)
```

Other special characters in the format strings are:

```
\n - line feed
\r - carriage return
\t - horizontal tab
\\ - backslash
%% - a literal % sign
```

`--streaming [ =yes | no | force | ext ]`

Uses streaming in file transfer, if server supports it. Files smaller than *buffer\_size\_bytes* are not transferred using streaming. Use *force* with small files. Default: *no*

Use *ext* with z/OS hosts to enable direct MVS dataset access. Use this option only when the file transfer is mainly used for mainframe dataset transfers, as it can slow down the transfer of small files in other environments.

The `--streaming=ext` option requires also the `--checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

An alternative way to activate extended streaming is to define `SSH_SFTP_STREAMING_MODE=ext` and `SSH_SFTP_CHECKSUM_MODE=no` as environment variables.

`--tcp-connect-timeout=VALUE`

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter the timeout value as a positive number. Value 0 (zero) disables this feature and the default system TCP timeout will be used, instead.

`--sunique`

Stores files with unique names. In case more than one of the transferred files have the same name, this feature adds a sequential number to the end of the repeated file name, for example: *file.name*, *file.name1*, and *file.name2*.

`-V, --version`

Displays program version and exits.

-h, --help, -?

Displays a short summary of command-line options and exits.

## Filename Support

Different operating systems allow different character sets in filenames. On Unix, some of the special characters are allowed in filenames, but on Windows, the following characters are not allowed:

```
\ / : * ? " < > |
```

When you use the **scp3** command to copy files with special characters (for example `unixfilename*?.txt`) from a Unix server to Windows, you need to provide the files with new names that are acceptable on Windows. Enter the commands in the following format:

```
$ scp3 user@unixserver:"unixfilename~*~?\".txt" windowsfilename.txt
```

The general rule is to follow your platform specific syntax when you enter filenames containing special characters as arguments to the **scp3** command.

SSH Tectia fully supports filenames containing only ASCII characters. Filenames containing characters from other character sets are not guaranteed to work.

## Using Wildcards

The **scp3** command supports `*` and `?` as wildcards.

The wildcards can be used both on the remote and the local side in the commands. The following example command will copy all text files (`*.txt`) from all subdirectories of directory `dir2` whose names begin with the prefix `data-` into the current local directory (`.`):

```
$ scp3 -r user@server:"dir2/data-*/*.txt" .
```

Note that on Unix, the characters `*` and `?` can appear also in the filenames. So it is necessary to use escape characters to distinguish the wildcards from the characters belonging to a filename. See more information in [the section called “Escaping Special Characters”](#).

## Escaping Special Characters

Some special characters that are used in filenames in different operating system, may have a special meaning in the SSH Tectia commands. Note also that the meaning can be different in various parts of the file transfer system.

In the **scp3** command, the following characters have a special meaning, and they need to be escaped in commands that take filenames as arguments:

`*` asterisk is a wildcard for any number of any characters

`?` question mark is a wildcard for any single character



"" quotation marks placed around strings that are to be taken 'as is'

\ backslash is an escape character on Unix

~ tilde is an escape character on Windows.

The escape character tells the **scp3** command to treat the next character "as is" and not to assume any special meaning for it. The escape character is selected according to the operating system of the local machine.

Note that the \ and ~ characters are special characters themselves, and if they are present in the filename, escape characters must be placed in front of them, too. Therefore, if you need to enter a filename containing \ in Unix or ~ in Windows to the **scp3** command, add the relevant escape character to it:

\\ on Unix

~~ on Windows

See the examples below to learn how the escape characters are used in the SSH Tectia **scp3** command, and how to enter filenames with special characters in different operating systems.

Examples of filenames in the **scp3** command:

The following filenames are valid in Unix, but they need escape characters in the commands:

```
file|name.txt
file-"name".txt
file?name.txt
file*name.txt
file\name.txt
file - name.txt
file~name.txt
```

When using the **scp3** command on Unix, in certain cases several escape characters are needed, as they escape one another. Enter the above mentioned filenames in the following formats:

```
file\|name.txt      or  "file|name.txt"
file-\ "name\".txt
file\\ \\?name.txt  or  "file\?name.txt"
file\\ \\*name.txt  or  "file\*name.txt"
file\\ \\name.txt   or  "file\\name.txt"
file\ -\ name.txt   or  "file - name.txt"
file~name.txt
```

Example commands on Unix:

```
$ scp3 user@server:file\\ \\*name.txt .
$ scp3 user@server:file\ -\ name.txt .
```

When using the **scp3** command on Windows, enter the above mentioned Unix filenames in the following formats:

```
"file|name.txt"
file-\ "name\".txt      (Note that Windows requires \ to escape the " character)
"file~?name.txt"
"file~*name.txt"
file~\name.txt
"file - name.txt"
file~~name.txt
```

The operating system interprets the quotation marks (") here so that the **scpg3** command receives the string without the quotation marks as a parameter.

Example commands on Windows:

```
> scpg3 user@server:"file~*name.txt" filename.txt
> scpg3 user@server:"file - name.txt" .
```

## Environment Variables

**scpg3** uses the following environment variables:

**SSH\_SFTP\_CHECKSUM\_MODE**=no|md5|md5-force|sha1|sha1-force|checkpoint

Defines the default checksum mode for **sftp3** and **scpg3** commands. Checksums are used to determine the point in the file where file transfer can be resumed if it gets interrupted.

no - checksums are not used; the file is always transferred from the beginning until EOF. This prevents staging in z/OS.

md5 – MD5 checksums are used

md5-force – MD5 checksums are forced

sha1 – SHA1 checksums are used

sha1-force – SHA1 checksums are forced

checkpoint – a separate checkpoint database is used.

**SSH\_SFTP\_OVERWRITE**=yes|no

If this variable is set to **yes** (default), the default behavior is to overwrite existing files. If set to **no**, the default behavior is not to overwrite existing files.

**SSH\_SFTP\_SHOW\_BYTE\_COUNT**=yes|no

If this variable is set to **yes**, the number of transferred bytes is shown after successful file transfer. Also the names of source and destination files are shown. The default is **no**.

**SSH\_SFTP\_STATISTICS**=yes|no|simple

If this variable is set to `yes` (default), normal progress bar is shown while transferring the file. If it is set to `no`, progress bar is not shown. If it is set to `simple` file transfer statistics are shown after the file has been transferred.

**SSH\_SFTP\_STREAMING\_MODE**=yes|no|ext

Defines the default streaming mode to be used with **sftpg3** and **scpg3** commands.

`no` – streaming is not used.

`yes` – standard streaming is used.

`ext` – extended streaming is used.

## Exit Values

**scpg3** returns the following values based on the success of the operation:

```
0      Operation was successful.
1      Internal error.
2      Connection aborted by the user.
3      Destination is not a directory, but a directory was specified by the user.
4      Connecting to the host failed.
5      Connection lost.
6      File does not exist.
7      No permission to access file.
8      Undetermined error from sshfilexfer.
11     Some non-fatal errors occurred during a directory operation.
101    Wrong command-line arguments specified by the user.
```

## Examples

Copy files from your local system to a remote Unix system:

```
$ scp3 localfile user@remotehost:/dst/dir/
```

Copy files from your local system to a remote Windows system:

```
$ scp3 localfile user@remotehost:/C:/dst/dir/
```

Copy files from a remote system to your local disk:

```
$ scp3 user@remotehost:/src/dir/srcfile /dst/dir/dstfile
```

Copy files from one remote system to another using connection profiles defined in the `ssh-broker-config.xml` file:

```
$ scp3 profile1:/src/dir/srcfile profile2:/dst/dir/dstfile
```

## sftpg3

sftpg3 -- Secure Shell file transfer client - Generation 3

### Synopsis

```
sftpg3 [options...]  
[ profile | [user@] host [#port] ]
```

### Description

**sftpg3** (**sftpg3.exe** on Windows) is an FTP-like client that can be used for secure file transfer over the network. **sftpg3** launches **ssh-broker-g3** to provide a secure transport using the Secure Shell version 2 protocol. **ssh-broker-g3** will ask for passwords or passphrases if they are needed for authentication. **sftpg3** uses the configuration specified in the `ssh-broker-config.xml` file.

When started interactively, **sftpg3** displays a prompt where the SFTP commands can be entered. It is also possible to start **sftpg3** non-interactively with a batch file that contains the commands to be run. For information on the available commands, see [the section called “Commands”](#).

**sftpg3** has two connection end points, local and remote, and both of them can be connected to other hosts than the SFTP client host. If started without arguments, the local end point is connected to the filesystem of the SFTP client host and the remote end point is unconnected. The connected host(s), with the exception of the SFTP client host, must be running a Secure Shell version 2 server with the **sftp-server** (or **sft-server-g3**) subsystem enabled. SSH Tectia Server has **sft-server-g3** enabled by default.

The remote connection end point can be given directly as an argument to the **sftpg3** command or it can be given with the **open** SFTP command after **sftpg3** has started. The local connection end point can be given with the **lopen** SFTP command.

When connecting, you can give either the name of a connection profile defined in the `ssh-broker-config.xml` file (*profile*) or the IP address or DNS name of the remote host, optionally with the remote username and the port of the Secure Shell server ( [*user@*] *host* [*#port*]). If no username is given, the local username is assumed. If no port is given, the default Secure Shell port 22 is assumed.



### Note

When entering a connection profile in **sftpg3**, note that SSH Tectia Client deduces the meaning of the argument differently depending on its format. If there is an @ sign in the given attribute value, SSH Tectia Client always interprets it to be `<username@hostname>`, i.e. not a profile.

Also, if there are dots in a profile name (for example `host.x.example.com`, the dots need to be escaped on command line. On Unix, enter `host\.x\.example\.com`, instead. On Windows, enter `host~.x~.example~.com`, instead. Otherwise the profile name is taken as a host name and the current local user name is used for logging in.

For information on special characters in file names, see [the section called “Filename Support”](#).

## Options

The following options are available:

`-b buffer_size_bytes`

Defines the maximum buffer size for one read/write request (default: 32768 bytes).

`-B batch_file`

Uses batch mode and executes SFTP commands from *batch\_file*. The file can contain any allowed SFTP commands. For a description of the commands, see [the section called “Commands”](#).

Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

`-C`

Disables compression from the current connection.

`+C`

Enables zlib compression for this particular connection.

`-c, --ciphers=LIST`

Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

Enter `help` as the value to view the currently supported cipher names.

`-D, --debug=LEVEL`

Sets the debug level. *LEVEL* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.



### Note

Option `-D` only applies on Unix. On Windows, instead of this command-line tool, use the Connection Broker debugging options `-D, -l`.



### Note

The debug level can be set only when the **sftpg3** command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

`-i FILE`

Defines that private keys defined in the identification file are used for public-key authentication.

`-K, --identity-key-file=FILE`

Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple `-K` options on command line.

`-N max_requests`

Defines the maximum number of read/write requests sent in parallel (default: 10).

`-P port`

Connects to this Secure Shell port on the remote machine (default: 22).

`-q, --quiet`

Suppresses the printing of error, warning, and informational messages.

`-v, --verbose`

Uses verbose mode (equal to `-D 2`).

`+w, --try-empty-password`

Tries an empty password.

`--allowed-authentications=METHODS`

Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. Enter `help` as the value to view the currently supported authentication methods.

`--compressions=METHODS`

Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter `help` as the value to view the currently supported compression methods.

`--exclusive`

Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

`--fips`

Performs the checksums using the FIPS cryptographic library.

`--identity=ID`

Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

`--identity-key-hash=ID`

Defines the private key used in user authentication with the corresponding public key hash.

`--identity-key-id=ID`

Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

`--keep-alive=VALUE`

Defines how often keep-alive messages are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

`--macs=LIST`

Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

```
--mac hmac-shal-96,hmac-md5,hmac-md5-96
```

Enter `help` as the value to view the currently supported MAC names.

`--password=PASSWORD | file://PASSWORDFILE | extprog://PROGRAM`

Sets the user password or passphrase that the client will send as a response to an authentication method requesting a password or passphrase (hereafter: password). This can be used also with password-protected certificates and public-keys.

The `PASSWORD` can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (`--password=file://PASSWORDFILE`), or entering a path to a program or script that outputs the password (`--password=extprog://PROGRAM`).

When using the `extprog://` option to refer to a shell script, make sure the script also defines the user's shell, and outputs the actual password. Otherwise the executed program fails, because it does not know what shell to use for the shell script. For example, if the password string is defined in a file named `my_password.txt`, and you want to use the bash shell, include these lines in the script:

```
#!/usr/bash
cat /full/pathname/to/my_password.txt
```



## Caution

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

`--plugin-path=PATH`

Sets plugin path to `PATH`. This is only used in the FIPS mode.

`--tcp-connect-timeout=VALUE`

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter a timeout value as a positive number. Value 0 (zero) disables this feature and the default system TCP timeout will be used, instead.

`-V, --version`

Displays program version and exits.

`-h, --help`

Displays a short summary of command-line options and exits.

## Commands

When **sftpg3** is ready to accept commands, it will display the prompt `sftp>`. The user can then enter any of the following commands:

`! [command] [arguments]`

Invokes an interactive shell on the local machine. if a *command* is given, it is used as the command to be executed. Optional *arguments* can be given depending on the command.

`append [-u, --unlink-source] [--streaming] [--force-lower-case] [--statistics] [--statistics-format] [--progress-display] [--progress-line-format] [--progress-line-interval] srcfile [dstfile]`

Appends the specified local file to the remote file. No globbing can be used.

Options:

`-u, --unlink-source`

Removes the source file after file transfer.

`--streaming [ =yes | no | force | ext ]`

Uses streaming in file transfer if the server supports it. Files smaller than *buffer\_size\_bytes* are not transferred using streaming. Use *force* with small files. Default: *no*

Use *ext* with z/OS hosts to enable direct MVS dataset access. Use this option only when the file transfer is mainly used for mainframe dataset transfers, as it can slow down the transfer of small files in other environments.

The `--streaming=ext` option requires also the `--checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

An alternative way to activate extended streaming is to define `SSH_SFTP_STREAMING_MODE=ext` and `SSH_SFTP_CHECKSUM_MODE=no` as environment variables.

`--force-lower-case`

Destination filename will be converted to lowercase characters.

The semantics of options `--statistics`, `--statistics-format`, `--progress-display`, `--progress-line-format`, and `--progress-line-interval` are the same as with **get**.

`ascii [-s] [remote_nl_conv] [local_nl_conv]`

Command **ascii** sets the transfer mode to ASCII.



For transfers between SSH Tectia on z/OS and other hosts, this also enables automatic ASCII-EBCDIC conversion. Default conversion is between codesets ISO8859-1 and IBM-1047. Files are transferred using the `LINE` format. The `site` and `lsite` commands can be used to change the values.

If you enter the `ascii` command with any options, it does not set the transfer mode to ASCII, but affects the newline conventions used in the transferred files.

#### Options:

`-s`

Shows the current newline convention. The line delimiters used in different systems are:

```
dos:    LFNL (\r\n, 0x0d0a)
mac:    LF  (\r, 0x0d)
mvs:    NL  (\n, 0x0a)
unix:   NL  (\n, 0x0a)
```

`remote_nl_conv local_nl_conv`

This syntax can be used to define the remote and local newline conventions. The `local_nl_conv` option operates on the local end, but notice that usually the correct local newline convention is already compiled in.

You can either set hints of the newline conventions for the underlying transfer layer, which by default tries to use the actual newline convention given by the server — or you can force the newline mode.

To set hints of the newline conventions, use these values in the `remote_nl_conv` and `local_nl_conv` options: `dos`, `unix`, and `mac`. These settings will be used if the remote SSH server does not automatically provide any newline information to the SFTP client. For example:

```
sftp> ascii
File transfer mode is now ascii.
sftp> ascii unix dos
Newline conventions updated.
```

To force the newline conventions, use these values: `force-dos`, `force-unix`, and `force-mac`. These settings force the newline mode irrespective of what the remote SSH server suggests to the SFTP client.

```
sftp> ascii
File transfer mode is now ascii.
sftp> ascii force-unix force-dos
Newline conventions updated.
```

You can also set either one of the options to `ask`, which will cause **sftpg3** to prompt you for the newline convention when needed.

`auto`

File transfer mode will be selected automatically from the file extension.

**binary**

Files will be transferred in binary mode.

**break**

Interrupts batch file execution. Batch file execution can be continued with the **continue** command.

**bye**

Quits the application.

**cd *directory***

Changes the current remote working directory.

**chmod** [-R] [-f] [-v] *OCTAL-MODE* [*file...*]

,

**chmod** [-R] [-f] [-v] [*ugo*][*a*] [*+-=*] [*rxws*] [*file...*]

Sets file permissions of the specified file or files to the bit pattern *OCTAL-MODE* or changes permissions according to the symbolic mode [*ugo*][*a*] [*+-=*] [*rxws*]. Only one symbolic mode combination is supported.

**Options:**

-R

Recursively changes files and directories.

-f

Uses silent mode (error messages are suppressed).

-v

Uses verbose mode (lists every file processed).

**close**

Closes the remote connection.

**continue**

Continues interrupted batch file execution.

**debug** [ *disable* | *no* | *debuglevel* ]

Disables or enables debug. With *disable* or *no*, debugging is disabled. Otherwise, sets *debuglevel* as debug level string, as per command-line option -D.

**digest** [-H, --hash] [-o, --offset] [-l, --length] *file*

Calculates MD5 or SHA-1 digest over file data.

**Options:**

-H, --hash= [ *md5* | *sha1* ]

Use *md5* or *sha1* hash algorithm (default: *md5*).

`-o, --offset=OFFSET`

Start reading from file offset *OFFSET*.

`-l, --length=LENGTH`

Read *LENGTH* bytes of file data.

`echo Text to be echoed.`

Echo the text. This command can be used for example in batch mode to print text into batch logs.

`get [-p, --preserve-attributes] [-u, --unlink-source] [-I, --interactive] [--overwrite] [--checksum] [-W, --whole-file] [--checkpoint] [--streaming] [--force-lower-case] [--prefix] [--statistics] [--statistics-format] [--progress-display] [--progress-line-format] [--progress-line-interval] [--max-depth=] file...`

Transfers the specified files from the remote end to the local end. By default, directories are recursively copied with their contents, but this is configurable in the Connection Broker configuration with the SFTP compatibility mode setting (`sftpg3-mode` in `ssh-broker-config.xml`). To view the currently set SFTP compatibility mode, run command:

```
sftp> help get
```

The currently set compatibility mode is shown in the beginning of the help for command **get**.

The SFTP compatibility mode options are:

`tectia`

The **sftpg3** client transfers files recursively from the current directory and all its subdirectories.

`ftp`

The `get` command is executed as `sget` meaning that it transfer a single file, and no subdirectories are copied.

`openssh`

Only regular files and symbolic links from the specified directory are copied, and no subdirectories are copied. Otherwise the semantics of the **get** command are unchanged.

Options:

`-p, --preserve-attributes`

Preserves the file permissions and the timestamps when both the source and the destination are on Unix filesystems (including z/OS USS). Preserves the timestamps but not the file permissions, if either one, the source or the destination is on Windows. If the destination is on z/OS MVS, none will be preserved.

`-u, --unlink-source`

Removes the source file after file transfer. Also directories are removed, if they become empty (move mode).

`-I, --interactive`

Prompts whether to overwrite an existing destination file (does not work with batch mode).

`--overwrite [ =yes | no ]`

Decides whether to overwrite existing destination file(s) (default: *yes*).

`--checksum [ =yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]`

Uses MD5 or SHA-1 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than *buffer\_size\_bytes* are not checked. Use *md5-force* or *sha1-force* with small files (default: *yes*, i.e. use MD5 checksums). Use checkpointing when transferring large files one by one.

`-W, --whole-file`

Does not try incremental checks. By default (if this option is not given), incremental checks are tried. This option can only be used together with the `--checksum` option.

`--checkpoint=s<seconds>`

Time interval between checkpoint updates (default: *10* seconds). This option can only be used when `--checksum=checkpoint`.

`--checkpoint=b<bytes>`

Byte interval between checkpoint updates (default: *10 MB*). This option can only be used when `--checksum=checkpoint`.

`--streaming [ =yes | no | force | ext ]`

Uses streaming in file transfer if the server supports it. Files smaller than *buffer\_size\_bytes* are not transferred using streaming. Use *force* with small files. Default: *yes*

Use *ext* with z/OS hosts to enable direct MVS dataset access. Use this option only when the file transfer is mainly used for mainframe dataset transfers, as it can slow down the transfer of small files in other environments.

The `--streaming=ext` option requires also the `--checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

An alternative way to activate extended streaming is to define `SSH_SFTP_STREAMING_MODE=ext` and `SSH_SFTP_CHECKSUM_MODE=no` as environment variables.

`--force-lower-case`

Destination filename will be converted to lower case characters.

`--max-depth=VALUE`

Defines whether directories are copied recursively. The value can be:

0 - unlimited recursion, directories are recursively copied with their contents

1 - copies files from the specified directory only, not from subdirectories

2-n - copies files recursively from the specified number of directory levels. Here *n* means the system-specific maximum.

This command line option overrides the recursion depth set in the Connection Broker configuration with element `sftpg3-mode` and/or the setting made using environment variable `SSH_SFTP_CMD_GETPUT_MODE`.

`--prefix=PREFIX`

Adds prefix `PREFIX` to filename during the file transfer. The prefix is removed after the file has been successfully transferred.

`--statistics [ =no | yes | simple | bytes ]`

Chooses the style of the statistics to be shown after a file transfer operation. The options mean:

`no` - no statistics will be shown. This is the default.

`yes` - detailed statistics will be created. You can configure the contents with the `statistics-format` option. The default statistics contents are:

Default settings:	Render for example this:
"Source: %c:%g\n"	user@host1#22:/path/to/source/file
"Source parameters: %e\n"	X=TEXT, C=ISO8859-1,D=IBM.1047
"Destination: %C:%G\n"	user@host2#22:/path/to/destination/file
"Destination parameters: %E\n"	NONE
"File size: %s bytes\n"	123456 bytes
"Transferred: %t bytes\n"	123456 bytes
"Rate: %RB/s\n"	345kiB/s
"Start: %xy-%xt-%xd %xh:%xm:%xs\n"	2008-12-19 13:10:56
"Stop: %Xy-%Xt-%Xd %Xh:%Xm:%Xs\n"	2008-12-19 13:23:30
"Time: %y\n"	00:12:34

`simple` - simple one-line statistics will be created. You can configure the contents with the `statistics-format` option. The default statistics contents are:

Default settings:	Render for example this:
"%f   %TB   %RB/s   TOC: %z\n"	file   2.8kiB   72kiB/s   TOC: 00:00:38

`bytes` - basic statistics reporting the transferred bytes will be created. You can configure the contents with the `statistics-format` option. The default statistics contents are:

Default settings:	Render for example this:
"Transferred %t bytes, file: '%f' -> '%F'\n"	12345 bytes, file: 'file1' -> 'file2'

`--statistics-format= FORMAT_STRING`

Chooses the format and the contents of the statistics. Use this option when `--statistics=yes|simple|bytes`. Select the contents for the statistics using the following definitions:

```

%c - source connection: user@host#port or profile
%g - /path/to/source/file
%f - source file name
%e - source parameters (file transfer and dataset parameters)
%C - destination connection: user@host#port or profile
%G - /path/to/destination/file
%F - destination file name
%E - destination parameters (file transfer and dataset parameters)
%s - file size in bytes
%S - file size as "XXyB" (B, kiB, MiB or GiB)
%t - transfer size in bytes
%T - transfer size as "XXyB" (B, kiB, MiB or GiB)
%p - transfer percentage
%q - transfer rate in bit/s
%Q - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r - transfer rate in bytes/s
%R - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%D* - current date
%x* - start date
%X* - end date
%y - elapsed time
%Y - time remaining
%z - ETA or TOC, if transfer has finished
%Z - string "ETA" or "TOC", if transfer has finished

```

Where \* is one of the following:

```

h - hours (00-23)
m - minutes (00-59)
s - seconds (00-59)
f - milliseconds (0-999)
d - day of the month (1-31)
t - month (1-12)
y - year (1970-)

```

Other special characters in format strings are:

```

\n - line feed
\r - carriage return
\t - horizontal tab
\\ - backslash

```

`--progress-display [=no | bar | line ]`

Chooses the mode of displaying the progress during a file transfer operation. The default is `bar`, which shows a progress bar. Option `line` shows the progress information according to the settings made in the `--progress-line-format` option.

`--progress-line-format=FORMAT_STRING`

Chooses what information will be shown on the progress line. Use this option when `--progress-display=line`. Select the contents for the progress line using the following definitions:

```

%c - source connection: user@host#port or profile
%g - /path/to/source/file
%f - source file name
%e - source parameters (file transfer and dataset parameters)
%C - destination connection: user@host#port or profile
%G - /path/to/destination/file
%F - destination file name
%E - destination parameters (file transfer and dataset parameters)
%s - file size in bytes
%S - file size as "XXyB" (B, kiB, MiB or GiB)
%t - transfer size in bytes
%T - transfer size as "XXyB" (B, kiB, MiB or GiB)
%p - transfer percentage
%q - transfer rate in bit/s
%Q - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r - transfer rate in bytes/s
%R - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%D* - current date
%x* - start date
%X* - end date
%y - elapsed time
%Y - time remaining
%z - ETA or TOC, if transfer has finished
%Z - string "ETA" or "TOC", if transfer has finished

```

Where \* is one of the following:

```

h - hours (00-23)
m - minutes (00-59)
s - seconds (00-59)
f - milliseconds (0-999)
d - day of the month (1-31)
t - month (1-12)
y - year (1970-)

```

Other special characters in format strings are:

```

\n - line feed
\r - carriage return
\t - horizontal tab
\\ - backslash

```

`--progress-line-interval=seconds`

Defines how often the progress information is updated in line mode. The interval is given in seconds, and the default is 60 seconds.

`getext`

Displays the extensions that will be ASCII in the auto transfer mode.

`lappend [options...] srcfile [dstfile]`

Same as **append**, but appends the specified remote file to the local file.

`lcd directory`

Changes the current local working directory.

`lchmod [-R] [-f] [-v] OCTAL-MODE [file...]`

,

`lchmod [-R] [-f] [-v] [ugoa] [+=-] [rwx] [file...]`

Same as **chmod**, but operates on local files.

`lclose`

Closes the local connection.

`ldigest [-H, --hash] [-o, --offset] [-l, --length] file`

Same as **digest**, but operates on local files.

`lls [-R] [-l] [-S] [-r] [-p] [-z|+z] [file...]`

Same as **ls**, but operates on local files.

`llsroots`

Same as **lsroots**, but operates on local files (when the local end has been opened to a VShell server).

`lmkdir directory`

Same as **mkdir**, but operates on local files.

`lopen hostname | -l`

Tries to connect the local end to the host *hostname*. If this is successful, **lls** and friends will operate on the filesystem on that host.

Options:

`-l`

Connects the local end to the filesystem of the SFTP client host (which does not require a server).

This is also the default state when no **lopen** commands have been given.

`lpwd`

Prints the name of the current local working directory.

`lreadlink path`

Same as **readlink**, but operates on local files.

`lrename oldfile newfile`

Same as **rename**, but operates on local files.

`lrm [options...] file...`

Same as **rm**, but operates on local files.

`lrmkdir directory`

Same as **rmdir**, but operates on local files.



`ls [-R] [-l] [-S] [-r] [-p] [-z | +z] [file...]`

Lists the names of files on the remote server. For directories, contents are listed. If no arguments are given, the contents of the current working directory are listed.

Options:

`-R`

Directory trees are listed recursively. By default, subdirectories of the arguments are not visited.

`-l`

Permissions, owners, sizes and modification times are also shown (long format).

`-S`

Sorting is done based on file sizes (default: alphabetical sorting).

`-r`

The sort order is reversed.

`-p`

Only one page of listing is shown at a time.

`-z`

The client generates the long output (alias for option `-l`).

`+z`

The long output supplied by the server is used, if available.

`lsite [ none | name1=value1 name2=value2... ]`

Same as **site**, but operates on local files and datasets.

`lsroots`

Dumps the virtual roots of the server. (This is a VShell extension. Without this you cannot know the filesystem structure of a VShell server.)

`lsymlink targetpath linkpath`

Same as **symlink**, but operates on local files.

`mget [options...] file...`

Synonymous to **get**.

`mkdir directory`

Tries to create the directory specified in *directory*.

`mput [options...] file...`

Synonymous to **put**.

`open hostname [-l]`

Tries to connect the remote end to the host *hostname*.

**Options:**`-l`

Connects the remote end to the filesystem of the SFTP client host (which does not require a server).

`pause [seconds]`

Pauses batch file execution for *seconds* seconds, or if *seconds* is not given until **ENTER** is pressed.

`put [options...] file...`

Transfers the specified files from the local end to the remote end. Options and semantics are the same as for **get**.

`pwd`

Prints the name of the current remote working directory.

`quit`

Quits the application.

`readlink path`

Provided that *path* is a symbolic link, shows where the link is pointing to.

`rename oldfile newfile`

Tries to rename the *oldfile* to *newfile*. If *newfile* already exists, the files are left intact.

`rm [-I, --interactive] [-r, --recursive] file...`

Tries to delete a file or directory specified in *file*.

**Options:**`-I, --interactive`

Prompts whether to remove a file or directory (does not work with batch mode).

`-r, --recursive`

Directories are removed recursively.

`rmdir directory`

Tries to delete the directory specified in *directory*. This command removes the directory only if it is empty and has no subdirectories.

`set [ defaults | option1=value1 option2=value2... ]`

Sets the default values for various parameters. The **set** command takes the following options:

`defaults`

Sets the parameters to be system defaults.

`checksum [ =md5 | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]`

Uses MD5 or SHA-1 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than *buffer\_size\_bytes* are not checked. Use

`md5-force` or `sha1-force` with small files. The default is `md5` (in z/OS the default is `no`). Use checkpointing when transferring large files one by one.

`compatibility-mode [ =tectia | ftp | openssh ]`

Defines what mode of recursiveness is used in the file transfer:

`tectia`

The **sftpg3** client transfers files recursively from the current directory and all its subdirectories. This is the default mode.

`ftp`

A single file is transferred, and no subdirectories are copied.

`openssh`

Only regular files and symbolic links from the specified directory are copied, and no subdirectories are copied.

`compressions [ =none | zlib ]`

Defines whether compression is used in file transfer:

`none`

Compression is not used. This is the default.

`zlib`

Enables zlib compression in file transfer.

`exit-value=VALUE`

Defines the exit value of **sftpg3** in batch mode in case an error occurred. The value must be between 0 and 255. If `exit-value` is set to something else than 0, batch execution terminates when the first error occurs. The default value is 0.

`overwrite [ =yes | no ]`

Decides whether to overwrite existing destination file(s) (default: `yes`).

`progress-display [ =bar | line | no ]`

Chooses the mode of displaying the progress during a file transfer operation. The default is `bar`, which shows a progress bar. Option `line` shows the progress information according to the settings made with the `progress-line-format` option. Option `no` disables progress display.

`progress-line-format=FORMAT_STRING`

Chooses what information will be shown on the progress line. Use this option when `--progress-display=line`. See the definitions of contents options in command: `get --progress-line-format`.

`progress-line-interval=seconds`

Defines how often the progress information is updated in line mode. The interval is given in seconds, and the default is 60 seconds.

`statistics-display [ =no | yes | simple | bytes ]`

Chooses the style of the statistics to be shown after a file transfer operation (default: `no`). See the options described for command: `get --statistics`.

`statistics-format=FORMAT_STRING`

Chooses the format and contents of the statistics. Use this command when `statistics-display=yes|simple|bytes`. See the definitions of contents options in command: `get --statistics-format`

`streaming [ =no | yes | force | ext ]`

Uses streaming in file transfer if the server supports it. Files smaller than `buffer_size_bytes` are not transferred using streaming. Use `force` with small files. Default: `no`

Use `ext` with z/OS hosts to enable direct MVS dataset access. Use this option only when the file transfer is mainly used for mainframe dataset transfers, as it can slow down the transfer of small files in other environments.

The `streaming=ext` option requires also the `checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

An alternative way to activate extended streaming is to define `SSH_SFTP_STREAMING_MODE=ext` and `SSH_SFTP_CHECKSUM_MODE=no` as environment variables.

`setext [extension...]`

Sets the file extensions that will be ASCII in the auto transfer mode. Normal `zsh-fileglob` regexps can be used in the file extensions.

`setperm fileperm [:dirperm]`

Sets the default file or directory permission bits for upload. (Prefix `fileperm` with `p` to preserve permissions of existing files or directories.)

`sget [options...] srcfile [dstfile]`

Transfers a single specified file from the remote end to the local end under the filename defined with `dstfile`. Directories are not copied. No wildcards can be used. Options are the same as for **get**.

`site [ none | name1=value1 name2=value2... ]`

Sets the file and dataset parameters for the remote host. Parameters can be entered either one by one, or several parameters can be delimited by spaces or commas. Both long parameters and abbreviations can be used. When run without arguments, the **site** command outputs the list of entered parameters. Setting `none` resets all parameters.

The available parameters are:

- `A|transfer_translate_dsn_templates=TEMPLATES`
- `automount=YES|NO|IMMED`

- AUTOMount
- autorecall=YES|NO
- AUTOREcall
- B|BLKsize|BLOCKSize=SIZE
- BLocks
- C|transfer\_codeset=CODESET
- CONddisp=CATLG|UNCATLG|KEEP|DELETE
- CYlinders
- D|transfer\_file\_codeset=CODESET
- DAtaClass|dataclas=CLASS
- dataset\_sequence\_number=NUMBER
- DEfer|defer=YES|NO
- E|transfer\_translate\_table=TABLE
- expiry\_date=YYDDD/YYYYDDD
- F|transfer\_format=LINE|STREAM|RECORD
- file\_status=NEW|MOD|SHR|OLD
- FIxrecfm=LENGTH
- I|transfer\_line\_delimiter=UNIX|MVS/DOS/MAC
- J|transfer\_file\_line\_delimiter=UNIX|MVS/DOS/MAC
- keylen=LENGTH
- keyoff=OFFSET
- L|size=SIZE
- label\_type=SL|NSL|SUL|LTM|AL|AUL
- like=LIKE
- M|Directory|directory\_size=SIZE
- MGmtclass|mgmtclas=CLASS

- NOAUTOMount
- NOAUTOREcall
- NORmdisp=CATLG|UNCATLG|KEEP|DELETE
- NOTRAILIngblanks
- NOTRUNcate
- O|RECfm=RECFM
- P|profile=PROFILE
- PRImary|primary\_space=SPACE
- R|LRecl=LENGTH
- RETpd|retention\_period=DAYS
- SECondary|secondary\_space=SPACE
- space\_unit=BLKS|TRKS|CYLS|AVGRECLEN
- space\_unit\_length=LENGTH
- STorclass|storclas=CLASS
- svc99\_text\_units=STRING
- T|type=PS|GDG|PO|PDS|POE|PDSE|HFS|VSAM|ESDS|KSDS|RRN|PREFIX|ALIAS
- TRACKs
- trailing\_blanks=YES|NO
- TRAILIngblanks
- TRUNcate
- U|record\_truncate=YES|NO
- UCount|unit\_count=NUMBER
- unit=UNIT
- unit\_parallel=YES|NO
- VCount|volume\_count=NUMBER
- volumes=VOL1+VOL2+...

- `X|transfer_mode=BIN/TEXT`

`sput [options...] srcfile [dstfile]`

Transfers a single specified file from the local end to the remote end under the filename defined with *dstfile*. Directories are not copied. No wildcards can be used. Options are the same as for **get**.

`sunique [on] [off]`

Stores files with unique names. If no option is specified, the command toggles the state of 'sunique'.

In case more than one of the transferred files have the same name, this feature adds a sequential number to the end of the repeated file name, for example: `file.name`, `file.name1`, and `file.name2`.

`symlink targetpath linkpath`

Creates symbolic link *linkpath*, which will point to *targetpath*.

`verbose`

Enables verbose mode (identical to the **debug 2** command). You may later disable verbose mode by **debug disable**.

`help [topic]`

If *topic* is not given, lists the available topics. If *topic* is given, outputs available online help about the topic.

`helpall`

Outputs available online help about all topics.

## Command Interpretation

**sftpg3** understands both backslashes (\) and quotation marks (""") on the command line. A backslash can be used for ignoring the special meaning of any character in the command-line interpretation. It will be removed even if the character it precedes has no special meaning.

Quotation marks can be used for specifying filenames with spaces.



### Note

Commands **get .** and **put .** will get or put every file in the current directory and possibly they overwrite files in your current directory.

**sftpg3** supports wildcard characters (also known as glob patterns) given to commands **chmod**, **lchmod**, **ls**, **lls**, **rm**, **lrm**, **get**, and **put**.

## Command-Line Editing (Unix)

On Unix, the following key sequences can be used for command-line editing:

**Ctrl-Space**

Set mark.

**Ctrl-A**

Go to the beginning of the line.

**Ctrl-B**

Move the cursor one character to the left.

**Ctrl-D**

Erase the character to the right of the cursor, or exit the program if the command line is empty.

**Ctrl-E**

Go to the end of the line.

**Ctrl-F**

Move the cursor one character to the right.

**Ctrl-H**

Backspace.

**Ctrl-I**

Tab.

**Ctrl-J**

Enter.

**Ctrl-K**

Delete the rest of the line.

**Ctrl-L**

Redraw the line.

**Ctrl-M**

Enter.

**Ctrl-N**

Move to the next line.

**Ctrl-P**

Move to the previous line.

**Ctrl-T**

Toggle two characters.

**Ctrl-U**

Delete the line.



**Ctrl-W**

Delete a region (the region's other end is marked with Ctrl-Space).

**Ctrl-X**

Begin an extended command.

**Ctrl-Y**

Yank deleted line.

**Ctrl-\_**

Undo.

**Ctrl-X Ctrl-L**

Lower case region.

**Ctrl-X Ctrl-U**

Upper case region.

**Ctrl-X Ctrl-X**

Exchange cursor and mark.

**Ctrl-X H**

Mark the whole buffer.

**Ctrl-X U**

Undo.

**Esc Ctrl-H**

Backwards word delete.

**Esc Delete**

Backwards word delete.

**Esc Space**

Delete extra spaces (leaves only one space).

**Esc <**

Go to the beginning of the line.

**Esc >**

Go to the end of the line.

**Esc @**

Mark current word.

**Esc A**

Go back one sentence.

**Esc B**

Go back one word.

**Esc C**

Capitalize current word.

**Esc D**

Delete current word.

**Esc E**

Go forward one sentence.

**Esc F**

Go forward one word.

**Esc K**

Delete current sentence.

**Esc L**

Change current word to lower case.

**Esc T**

Transpose words.

**Esc U**

Change current word to upper case.

**Delete**

Backspace.

## Filename Support

Different operating systems allow different character sets in filenames. On Unix, some of the special characters are allowed in filenames, but on Windows, the following characters are not allowed:

```
\ / : * ? " < > |
```

The **sftpg3** command-line tool (both as an interactive and in a batch file) follows the syntax and semantics of Unix shell command-line also on the Windows platform, except that the escape character is ~ (tilde).

When you transfer files that have special characters in the filename (for example `unixfilename*?.txt`) from a Unix server to Windows, you need to provide the files with new names that are acceptable on Windows.

The **sftpg3** command-line client includes two versions of the **get** command:

The **get** command can be used to transfer several files at the same time, but it is not possible to define target filenames. Note that if there are special characters in the filenames, you need to rename the files already on Unix so that the names are acceptable also on Windows.

The **sget** command is used to transfer one file at a time, and it allows you to define a new name for the destination file. Use it to make the name acceptable on Windows. The command sequence is as follows:

```
$ sftpg3  
  
sftp> open user@server  
  
sftp> sget "file*name.txt" windowsfilename.txt
```

## Escaping special characters

In the **sftpg3** command, the following characters have a special meaning, and they need to be escaped in commands that take filenames as arguments:

\* asterisk is a wildcard character for any number of any characters

? question mark is a wildcard for any single character

"" quotation marks are placed around strings that are to be taken 'as is'

\ backslash is an escape character on Unix

~ tilde is an escape character on Windows

The escape character tells the **sftpg3** command to treat the next character "as is" and not to assume any special meaning for it. The escape character is selected according to the operating system of the local machine.

Note that the \ and ~ characters are special characters themselves, and if they are present in the filename, an escape character must be placed in front of them, too. Therefore, if you need to enter a filename containing \ in Unix or ~ in Windows to any of the **sftpg3** commands, add the relevant escape character to it:

\\ on Unix

~~ on Windows

When a filename or part of a filename is placed within the quotation marks "", the **sftpg3** command interprets the quoted part 'as is', and none of the characters within the quote are interpreted as wildcards or as any other special characters.

However, on Unix a quotation mark " can also be part of a filename. If you need to enter the " character in a filename, you must add the escape character in front of it both on Unix and on Windows.

For example, to enter a file named file-"name".txt into a command on Windows, enter the following command:

```
sftp> sget "file-~"name~".txt" filename.txt
```

See the examples below to learn how the escape characters are used in the SSH Tectia **sftpg3** commands, and how to enter filenames with special characters in different operating systems.

Examples of filenames in the **sftpg3** commands:

The following filenames are valid in Unix, but they need escape characters in the commands:

```
file|name.txt
file-"name".txt
file?name.txt
file*name.txt
file\name.txt
file - name.txt
file~name.txt
```

When using the **sftpg3** command-line tool on Unix, enter the above mentioned filenames in the following formats:

```
file\|name.txt      or  "file|name.txt"
file-\ "name\".txt  or  "file-\ "name\".txt"
file\?name.txt      or  "file?name.txt"
file\*name.txt      or  "file*name.txt"
file\\name.txt      or  "file\\name.txt"
file\ -\ name.txt   or  "file - name.txt"
file~name.txt       or  "file~name.txt"
```

Example commands on Unix:

```
sftp> get "file*name.txt"
```

```
sftp> sget "file*name.txt" newfilename.txt
```

When using the **sftpg3** command on Windows, enter the above mentioned Unix filenames in the following formats:

```
file~|name.txt      or  "file|name.txt"
file~-~ "name~".txt or  "file~-~ "name~".txt"
file~?name.txt      or  "file?name.txt"
file~*name.txt      or  "file*name.txt"
file~\name.txt      or  "file\name.txt"
file~ ~~ name.txt   or  "file - name.txt"
file~~name.txt      or  "file~~name.txt"
```

Example command sequence on Windows:

```
> sftpg3 open user@server
```

```
sftp> get "file name.txt"
```

```
sftp> sget "file*name.txt" filename.txt
```

## Environment Variables

**sftpg3** uses the following environment variables:

**SSH\_SFTP\_BATCH\_FILE=FILE**

Defines the path to the batch file to be run when **sftpg3** is started. This can be used for example to perform a certain action before an interactive session is started.

**SSH\_SFTP\_CHECKSUM\_MODE=no|md5|md5-force|sha1|sha1-force|checkpoint**

Defines the default checksum mode for **sftpg3** and **scpg3** commands. Checksums are used to determine the point in the file where file transfer can be resumed if it gets interrupted.

**no** - checksums are not used; the file is always transferred from the beginning until EOF. This prevents staging in z/OS.

**md5** - MD5 checksums are used

**md5-force** - MD5 checksums are forced

**sha1** - SHA1 checksums are used

**sha1-force** - SHA1 checksums are forced

**checkpoint** - a separate checkpoint database is used.

**SSH\_SFTP\_CMD\_GETPUT\_MODE=tectia|ftp|openssh**

Defines the SFTP compatibility mode for transferring files. This setting affects the behaviour of the **get/mget/sget** and **put/mput/sput** commands and the recursion level used by the **sftpg3** client. This environment variable overrides the **sftpg3-mode** setting made in the **ssh-broker-config.xml** file. The variables are:

**tectia** - Commands **get** and **put** work as usually, while **sget** and **sput** allow you to define the destination file name. Directories are copied recursively together with their contents. This is the default mode.

**ftp** - Commands **get/put** are executed as **sget/sput** meaning that they transfer a single file; and commands **mget/mput** have recursion depth set to 1, meaning that they only transfer files from the specified directory, not from subdirectories.

**openssh** - Commands **get/put/mget/mput** behave alike, meaning that only regular files and symbolic links from the specified directory are transferred. No subdirectories are copied.

**SSH\_SFTP\_OVERWRITE=yes|no**

If this variable is set to **yes** (default), the default behavior is to overwrite existing files. If set to **no**, the default behavior is not to overwrite existing files.

**SSH\_SFTP\_SHOW\_BYTE\_COUNT=yes|no**

If this variable is set to **yes**, the number of transferred bytes is shown after successful file transfer. Also the names of source and destination files are shown. The default is **no**.

**SSH\_SFTP\_STATISTICS**=yes|no|simple

If this variable is set to *yes* (default), normal progress bar is shown while transferring the file. If it is set to *no*, progress bar is not shown. If it is set to *simple*, file transfer statistics are shown after the file has been transferred.

**SSH\_SFTP\_STREAMING\_MODE**=yes|no|ext

Defines the default streaming mode to be used with **sftpg3** and **scpg3** commands.

*no* – streaming is not used.

*yes* – standard streaming is used.

*ext* – extended streaming is used.

## Exit Values

**sftpg3** returns the following values based on the success of the operation:

```
0      Operation was successful.
1      Internal error.
2      Connection aborted by the user.
3      Destination is not a directory, but a directory was specified by the user.
4      Connecting to the host failed.
5      Connection lost.
6      File does not exist.
7      No permission to access file.
8      Undetermined error from sshfilexfer.
11     Some non-fatal errors occurred during a directory operation.
101    Wrong command-line arguments specified by the user.
```

In batch mode, **sftpg3** returns the value 0 only if no errors occurred during the execution. A failure to change the current working directory, a failure to establish a connection, or a connection loss during batch operation cause **sftpg3** to abort. Other errors are reported to *stderr* and the last error value is returned as the exit value of the **sftpg3** process.

## Examples

Open a **sftpg3** session with the remote end connected to the server defined in the connection profile *profile1* in the *ssh-broker-config.xml* file (the local end is initially connected to the filesystem of the SFTP client host):

```
$ sftpg3 profile1
```

Run **sftpg3** in batch mode:

```
$ sftpg3 -B batch.txt
```

Example contents of the batch file *batch.txt* are shown below. Non-interactive authentication methods are used and the server host keys have been stored beforehand:

```
lopen user@unixserver.example.com
open user@winserver.example.com
binary
lcd backup
cd c:/temp
get --force-lower-case Testfile-X.bin
lchmod 700 testfile-x.bin
quit
```

The example batch file opens the local end of the connection to a Unix server and the remote end to a Windows server, and sets the transfer mode to binary. It changes to local directory *backup* and remote directory *C:\Temp*, and copies a file from the remote directory to the local directory. The filename is changed to lower-case characters (*testfile-x.bin*). After transfer, the file permissions are changed to allow the user full rights and others no rights.

## ssh-translation-table

ssh-translation-table -- Secure Shell File Transfer Translation Table

### Synopsis

```
ssh-translation-table [options...]
[filename]
```

### Description

**ssh-translation-table** (**ssh-translation-table.exe** on Windows) is a utility program that can be used for generating template translation tables used in file transfer with the **scp3** or **sftp3** file transfer clients. Translation table is stored to *filename*. If *filename* is not given, translation table is displayed.

### Options

The following options are available:

**-b, --binary**

Use z/OS-specific binary file format.

**-f, --from=CODESET**

Specify the codeset of the data in the file when translation is done while reading from the file, or codeset of the data in transfer when translation is done while writing to the file (default: *ISO8859-1*). For example:

```
--from ISO8859-1
```

`-t, --to=CODESET`

Specify the codeset of the data in transfer when translation is done while reading from the file, or codeset of the data in the file when translation is done while writing to the file (default: *IBM-1047,swaplfnl*). For example:

```
--to IBM-1047
```

`-l, --list-charsets`

List available character sets. Note that all character sets are not single byte character sets. Only single byte character sets can be used.

`-D, --debug=LEVEL`

Sets the debug level. *LEVEL* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

`-h, --help`

Displays a short summary of command-line options and exits.

## File Transfer Translation Table

File transfer translation table is a simple text file containing two tables describing the character conversion. The first table is used for converting data while writing to the file and the second table is used for converting the data while reading from the file. The table itself is a simple list of 255 values represented as two hexadecimal values (from 00 to FF). The position of the value is the index for conversion. The first position, i.e. position 00, represents the converted value for byte value of 0.

The hexadecimal values in the tables are case-insensitive. So values 0a and 0A are the same. Also, it is possible to add comments into the file. The comment starts with character '#'. Everything after that until end of line is treated as comment and ignored. Also all white spaces are ignored.



### Note

Only single byte translations are supported with translation tables.

Here is an example translation table generated with command **ssh-translation-table**:

```
## SSH TRANSLATION TABLE FILE FORMAT VERSION 1.0
#####
#
# This file is an example translation table that can be used to
# translate data from 'ISO8859-1' to 'IBM-1047,swaplfnl' while reading
# from a file or from 'IBM-1047,swaplfnl' to 'ISO8859-1' while writing
# to a file.
#
# The format of translation table file is following:
#
# - White spaces are ignored.
# - Everything after '#' character until end of line is a comment
```





```
6465626663679E687471727378757677 #C
AC69EDEEEBEFECBF80FDFEFBFCBAAE59 #D
4445424643479C485451525358555657 #E
8C49CDCECBCFCCE170DDEDBDC8D8EDF #F

# EOF
```



## Note

When ICU libraries are used for generating ASCII to EBCDIC translation tables, `',swaplfnl'` must be added to EBCDIC codepage name so that ASCII newline character is correctly translated to EBCDIC newline character.

In order to create custom translation tables, first create a standard table and manually edit it to suite your needs.

Translation tables can be used with SSH Tectia file transfer clients and server. The translation can be performed either in the client or in the server. The translation table file must be available in the host that is performing the translation.

## Using Translation Tables with **sftpg3**

Translation table filename is specified using site parameter `TRANSFER_TRANSLATE_TABLE` or `E`. Since translation table filename can be part of file transfer advice string, `'/'` character must be encoded as `'%2f'`. Also, the parameter must be just one value without spaces. Space must be encoded as `'%20'`.



## Note

File transfer clients **sftpg3** and **scpg3** encode `'/'` internally as `'%2f'`.



## Note

If translation is performed in SSH Tectia **sftpg3** or **scpg3** client, server does not have to be SSH Tectia.

Here is an upload example, where translation is done in the **sftpg3** client:

```
$ sftpg3 user@example.com
sftp> lsite E=/path/to/ISO8859-1_to_IBM-1047,swaplfnl.txt
sftp> sput ISO8859-1_file.txt IBM-1047_file.txt
```

With the **lsite** command, translation is activated on the local end. Text file is translated from ISO8859-1 to IBM-1047 and transferred as a binary file.

## Note

Since translation tables only handle single byte conversions it is not possible to change newline convention from DOS to Unix or MVS with the translation table only.

The following example shows the download of a file:

```
$ sftpg3 user@example.com
sftp> lsite E=/path/to/ISO8859-1_to_IBM-1047,swaplfnl.txt
sftp> sget IBM-1047_file.txt ISO8859-1_file.txt
```

In this upload example, translation is done in the remote server:

```
$ sftpg3 user@example.com
sftp> ascii
sftp> site E=/path/to/IBM-1047,swaplfnl_to_ISO8859-1.txt
sftp> sput ISO8859-1_file.txt IBM-1047_file.txt
```

With the **site** command, translation is activated on the remote end. Text file is first transferred as text to remote end and then translated from ISO8859-1 to IBM-1047.

## Note

Since now the translation is done on the remote end, it is possible to first perform newline conversion on the client. Also newline conversion where the length of the data changes, like from DOS to UNIX, can be done in this case.

The following example shows the download of the file:

```
$ sftpg3 user@example.com
sftp> ascii
sftp> site E=/path/to/IBM-1047,swaplfnl_to_ISO8859-1.txt
sftp> sget IBM-1047_file.txt ISO8859-1_file.txt
```

## Using Translation Tables with scp3

Here is an upload example, where translation is done in the **scp3** client:

```
$ scp3 --src-site=E=/path/to/ISO8859-1_to_IBM-1047,swaplfnl.txt
ISO8859-1_file.txt user@example.com:IBM-1047_file.txt
```

With the **--src-site** option, translation is activated on the local end. Text file is translated from ISO8859-1 to IBM-1047 and transferred as a binary file.

The following example shows the download of the file:

```
$ scp3 --dst-site E=/path/to/ISO8859-1_to_IBM-1047,swaplfnl.txt
user@example.com:IBM-1047_file.txt ISO8859-1_file.txt
```

In this upload example, translation is done in remote server:

```
$ scp3 -a --dst-site=E=/path/to/IBM-1047,swaplnl_to_ISO8859-1.txt
ISO8859-1_file.txt user@example.com:IBM-1047_file.txt
```

With the `--dst-site` option, translation is activated on the remote end. Text file is first transferred as text to remote end and then translated from ISO8859-1 to IBM-1047.



### Note

Since now the translation is done on the remote end, it is possible to first perform newline conversion on the client. Also newline conversion where the length of the data changes, like from DOS to UNIX, can be done in this case.

The following example shows the download of the file:

```
$ scp3 -a --src-site=E=/path/to/IBM-1047,swaplnl_to_ISO8859-1.txt
user@example.com:IBM-1047_file.txt ISO8859-1_file.txt
```

## Using Translation Tables with OpenSSH

Here is an upload example, where translation is done in SSH Tectia Server:

```
$ sftp user@example.com
sftp> put ISO8859-1_file.txt
/ftadv:E=%2fpath%2fto%2fIBM-1047,swaplnl_to_ISO8859-1.txt/IBM-1047_file.txt
```

With `ftadv` string in destination file name, translation is activated on the remote end. Text file is first transferred to remote end and then translated from ISO8859-1 to IBM-1047.



### Note

When translation table information is given in `ftadv` string, `'/'` character must be encoded as `'%2f'`. Also, since the parameter must be just one value without spaces, space must be encoded as `'%20'`.

The following example shows the download of the file:

```
$ sftp user@example.com
sftp> get /ftadv:E=%2fpath%2fto%2fIBM-1047,
swaplnl_to_ISO8859-1.txt/IBM-1047_file.txt ISO8859-1_file.txt
```

## ssh-keygen-g3

ssh-keygen-g3 -- authentication key pair generator

### Synopsis

```
ssh-keygen-g3 [options...]
[key1 key2...]
```

## Description

**ssh-keygen-g3** (**ssh-keygen-g3.exe** on Windows) is a tool that generates and manages authentication keys for Secure Shell. Each user wishing to use a Secure Shell client with public-key authentication can run this tool to create authentication keys. Additionally, the system administrator can use this to generate host keys for the Secure Shell server.

By default, if no path for the key files is specified, the key pair is generated under the user's home directory (`$HOME/.ssh2` on Unix, `%APPDATA%\SSH\UserKeys` on Windows). If no filename is specified, the key pair is likewise stored under the user's home directory with such filenames as `id_dsa_1024_a` and `id_dsa_1024_a.pub`.

## Options

The following options are available:

`-1 file`

Converts a key file from the SSH1 format to the SSH2 format. Note: "1" is number one (not letter L).

`-7 file`

Extracts certificates from a PKCS #7 file.

`-b bits`

Specifies the length of the generated key in bits (default: `2048`).

`-B num`

Specifies the number base for displaying key information (default: `10`).

`-c comment`

Specifies a comment string for the generated key.

`-D file`

Derives the public key from the private key *file*.

`-e file`

Edits the specified key. Makes **ssh-keygen-g3** interactive. You can change the key's passphrase or comment.

`-F, --fingerprint file`

Dumps the fingerprint of the given public key. By default, the fingerprint is given in the SSH Babble format, which makes the fingerprint look like a string of "real" words (making it easier to pronounce). The output format can be changed with the `--fingerprint-type` option.

The following options can be also used to modify the behaviour of this option: `--fingerprint-type`, `--hash`, `--hostkeys-directory`, `--known-hosts`, `--rfc4716`.

`-F, --fingerprint <host id>`

Dumps the fingerprint of the locally stored host key identified with the given `<host id>`. The `<host id>` is a host name or string "host#port";.

The following options can be used to modify the behaviour of this option: `--fingerprint-type`, `--hash`, `--hostkeys-directory`, `--known-hosts`, `--rfc4716`.

`-H, --hostkey`

Stores the generated key pair in the default host key directory (`/etc/ssh2` on Unix, "C:\Program Files\SSH Communications Security\SSH Tectia\SSH Tectia Server" on Windows). Specify the `-P` option to store the private key with an empty passphrase.

`-i file`

Loads and displays information on the key *file*.

`-k file`

Converts a PKCS #12 file to an SSH2-format certificate and private key.

`-p passphrase`

Specifies the passphrase for the generated key.

`-P`

Specifies that the generated key will be saved with an empty passphrase.

`-q, --quiet`

Hides the progress indicator during key generation.

`-r file`

Adds entropy from *file* to the random pool. If *file* contains 'relatively random' data (i.e. data unpredictable by a potential attacker), the randomness of the pool is increased. Good randomness is essential for the security of the generated keys.

`-t [ dsa | rsa ]`

Selects the type of the key. Valid options are `dsa` (default) and `rsa`.

`-x file`

Converts a private key from the X.509 format to the SSH2 format.

`--append [ =yes | no ]`

Appends the keys. Optional values are `yes` and `no`. The default is `yes` to append.

`--copy-host-id <host id> <destination>`

Copies the host identity to the specified destination directory.

The following options can be used to modify the behaviour of this option: `--append`, `--hostkeys-directory`, `--known-hosts`, `--overwrite`.

If `--hostkey-file` is given, the file is treated as a normal host identity file used by the Connection Broker, and its contents will be copied to the destination directory.

`--delete-host-id <host id>`

Deletes the host key of the specified host id. The `<host id>` is a host name or string "host#port";.

The following options can be used to modify the behaviour of this option: `--host-key-file`, `--hostkeys-directory`, `--known-hosts`.

`--fingerprint-type [ =babble | babble-upper | pgp-2 | pgp-5 | hex | hex-upper ]`

Specifies the output format of the fingerprint. If this option is given, the `-F` option and the key filename must precede it. The default format is `babble`.

See [the section called “Examples”](#) for examples of using this option.

`--fips-mode`

Generates the key using the FIPS mode for the cryptographic library. In the FIPS mode, only DSA keys of 1024 bits and RSA keys of at least 512 bits can be generated, and the keys must have non-empty passphrases. By default (if this option is not given), the key is generated using the standard mode for the cryptographic library.

`--fips-crypto-dll-path PATH`

Specifies the location of the FIPS cryptographic DLL.

`--hash [ =md5 | sha1 ]`

Specifies the digest algorithm for fingerprint generation. Valid options are `md5` and `sha1`.

`--hostkey-file file`

When copying, uses the given file as the source host key, instead of autodetecting the location. When deleting, only deletes from the given location. If the specified file does not contain identities for the specified host, does nothing.

`--hostkeys-directory directory`

Specifies the directory for known host keys to be used instead of the default location.

`--import-public-key infile outfile`

Attempts to import a public key from `infile` and store it to `outfile` in SSH2 native format.

`--import-private-key infile outfile`

Attempts to import an unencrypted private key from `infile` and store it to `outfile` in SSH2 native private key format.

`--import-ssh1-authorized-keys infile outfile`

Imports an SSH1-style `authorized_keys` file `infile` and generates an SSH2-style authorization file `outfile`, and stores the keys from `infile` to generated files into the same directory with `outfile`.

`--known-hosts file`

Uses the specified known hosts file. Enables fetching fingerprints for hosts defined in an OpenSSH-style known-hosts file. Using this option overrides the default locations of known\_hosts files (/etc/ssh/ssh\_known\_hosts and \$HOME/.ssh/known\_hosts). Giving an empty string will disable known-hosts usage altogether.

`--overwrite [ =yes | no ]`

Overwrite files with the same filenames. The default is to overwrite.

`--rfc4716`

Displays the fingerprint in the format specified in *RFC4716*. The digest algorithm (hash) is md5, and the output format is the 16-bytes output in lowercase HEX separated with colons (:).

`--set-hostkey-owner-and-dacl file`

On Windows, sets the correct owner and DACL (discretionary access control list) for the host key *file*. This option is used internally when a host key is generated during SSH Tectia Server installation.

`-V`

Displays version string and exits.

`-h, --help, -?`

Displays a short summary of command-line options and exits.

## Examples

Create a 1024-bit RSA key pair using the cryptographic library in the FIPS mode and store the key pair in the default user key directory with filenames *newkey* and *newkey.pub*:

```
$ ssh-keygen-g3 --fips-mode -t rsa -b 1024 newkey
```

Convert an SSH1 key *oldkey* to SSH2 format:

```
$ ssh-keygen-g3 -1 oldkey
```

Display the fingerprint of a server host public key in SSH babble (default) format:

```
$ ssh-keygen-g3 -F hostkey.pub
Fingerprint for key:
xeneh-fyvam-sotaf-gutuv-rahih-kipod-poten-byfam-hufeh-tydym-syxex
```

Display the fingerprint of a server host public key in hex format:

```
$ ssh-keygen-g3 -F hostkey.pub --fingerprint-type=hex
Fingerprint for key:
25533b8c7734f6eb1556ea2ab4900d854d5d088c
```



## ssh-keyfetch

ssh-keyfetch -- Host key tool for the Secure Shell client

### Synopsis

```
ssh-keyfetch [options...]  
[host]
```

### Description

**ssh-keyfetch** (**ssh-keyfetch.exe** on Windows) is a tool that downloads server host keys and optionally sets them as known host keys for the Secure Shell client. It is typically used by the system administrator during the initial setup phase.

By default the host key is fetched from the server and saved in file `key_host_port.suffix` in the current directory.

### Options

The following options are available:

`-a, --set-trusted`

Instead of writing the public key to a file, add the public key as a known host key to the user-specific directory: `$HOME/.ssh2/hostkeys` (`%APPDATA%\SSH\HostKeys` on Windows). This option cannot be combined with `-C` or `-K`.



### Caution

When **ssh-keyfetch** is run with the `-a` option, it accepts the received host keys automatically without prompting the user. You should verify the validity of keys by verifying the key fingerprints after receiving them or you risk being subject to a man-in-the-middle attack.

To validate the host key, obtain the host key fingerprint from a trusted source (for example by calling the server administrator) and verify it against the output from command:

```
ssh-keygen-g3 --fingerprint <hostname>
```

`-A, --fetch-any`

Probe for and fetch either server public key or certificate.

`-C, --fetch-certificate`

Probe for and fetch the server certificate only.

`-d, --debug debug-level`

Enable debugging.

`-D, --debug-default`

Enable debugging with default level.

`-f, --filename-format nameformat`

Filename format for known host keys. Accepted values are *plain* and *hashed*. The default is *plain*.

`-F, --fingerprint-type [ =babble | babble-upper | pgp-2 | pgp-5 | hex | hex-upper ]`

Public key fingerprint type for fingerprints displayed in messages and log. Most popular types are *babble* (the SSH babble format) and *hex*. The default is *babble*. See also the option `--rfc4716`.

`-H, --hash [ =md5 | sha1 ]`

Specifies the digest algorithm for fingerprint generation. Valid options are *md5* and *sha1*.

`-K, --kex-key-formats typelist`

Explicitly specify the host-key types accepted in protocol key exchange. For experts only. See RFC 4253 for details.

`-l, --log`

Report successfully received keys in log format. The log format consists of one line per key, six fields per line. The fields are:

- `accept|save`
- `replace|append`
- `hostname`
- `ip-port`
- `user-id`
- `key-file-path`
- `fingerprint`

`-o, --output-file output-file`

Write result to *output-file*. A minus sign ("-") denotes standard output.

`-O, --output-directory output-dir`

Write result to *output-dir*. The default is the current directory.

`-p, --port port`

Server port (default: 22).

`-P, --fetch-public-key`

Probe for and fetch the server public key only. This is the default behaviour.

`-q, --quiet`

Quiet mode, report only errors.

`-R, --rfc4716`

Displays the public key fingerprints in the format specified in RFC 4716. The digest algorithm (hash) is md5, and the output format is the 16-bytes output in lowercase HEX separated with colons (:).

`-S, --proxy-url socks-url`

Specifies the SOCKS server to use.

`-t, --timeout timeout`

Connection timeout in seconds (default: 10 seconds).

`--append [ =yes | no ]`

Instead of appending a new host key, overwrite the existing known host keys for this host. Optional values are *yes* and *no*. The default is to append.

`-V, --version`

Displays version string and exits.

## Environment Variables

### SSH SOCKS\_SERVER

The address of the SOCKS server used by **ssh-keyfetch**.

## Examples

Connect to the server through a SOCKS proxy:

```
$ ssh-keyfetch -S socks://fw.example.com:1080/10.0.0.0/8 server.outside.example
Public key from server.outside.example:22 saved.
File: server.outside.example.pub
Fingerprint: xucar-bened-liryt-lumup-minad-tozuc-pesyp-vafah-mugyd-susic-guxix
```

Accept the server key as a known key for SSH Tectia Client and report in the more rigid log format:

```
$ ssh-keyfetch -a -l newhost
Accepted newhost 22 testuser /home/testuser/.ssh2/hostkeys/key_22_newhost.pub
xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuv-muled-sxyx
```

Accept the server key as a known key for SSH Tectia Client and store the key to global configuration `hostkeys` directory:

```
$ ssh-keyfetch -a --output-directory /etc/ssh2/hostkeys
Accepted newhost 22 testuser /etc/ssh2/hostkeys/key_22_anotherhost.pub
bydop-mulym-zegar-nybuvmuled-syxyx-xigad-hozuf-kykek-vogid-dumid
```

Accept the server key as a known key for SSH Tectia Client and use an uninformative hash as the filename for the stored known key:

```
$ ssh-keyfetch -f hashed -a newhost
Public key from newhost:22 accepted as trusted hostkey.
File:
/home/testuser/.ssh2/hostkeys/keys_420b23ca959ab165e52e117a90baa89d92ffc535
Fingerprint:
xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuvmuled-syxyx
```

Fetch the X.509 certificate of the server running in port 222 and display the content with **ssh-certview**:

```
$ ssh-keyfetch -C -p 222 -o - newhost | ssh-certview -
Certificate =
  SubjectName = <C=FI, O=SSH, OU=DEV, CN=newhost.ssh.com>
  IssuerName = <C=FI, O=SSH, CN=Sickle CA>
  SerialNumber= 24593438
  Validity =
    NotBefore = 2007 Sep 13th, 15:10:00 GMT
    NotAfter = 2008 Sep 12th, 15:10:00 GMT
  PublicKeyInfo =
    PublicKey =
      Algorithm = RSA
      Modulus n (1024 bits) :
...
  Fingerprints =
    MD5 = 3c:71:17:9b:c2:12:26:cf:96:27:fb:d7:a8:19:37:89
    SHA-1 =
    14:72:f3:0f:20:5e:75:ed:d2:c3:86:4b:69:45:00:47:ae:fe:31:64
```

This explicit key exchange type list is equivalent to specifying option **-A**:

```
$ ssh-keyfetch -K ssh-rsa,ssh-dss,x509v3-sign-rsa,x509v3-sign-dss newhost
Public key from newhost:22 saved.
File: key_newhost_22.pub
Fingerprint:
xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuvmuled-syxyx
```

## ssh-cmpclient-g3

ssh-cmpclient-g3 -- CMP enrollment client

## Synopsis

```
ssh-cmpclient-g3 command [options] access [name]
```

Where command is one of the following:

```
INITIALIZE psk|racerts keypair template
ENROLL certs|racerts keypair template
UPDATE certs [keypair]
POLL psk|certs|racerts id

RECOVER psk|certs|racerts template
REVOKE psk|certs|racerts template
TUNNEL racerts template
```

Most commands can accept the following options:

```
-B          Perform key backup for subject keys.
-o prefix   Save result into files with prefix.
-O filename Save the result into the specified file.
            If there is more than one result file,
            the remaining results are rejected.
-C file     CA certificate from this file.
-S url      Use this SOCKS server to access the CA.
-H url      Use this HTTP proxy to access the CA.
-E          PoP by encryption (CA certificate needed).
-v num      Protocol version 1|2 of the CA platform. Default is 2.
-y          Non-interactive mode. All questions answered with 'y'.
-N file     Specifies a file to stir to the random pool.
```

The following identifiers are used to specify options:

```
psk      -p refnum:key (reference number and pre-shared key)
          -p file (containing refnum:key)
          -i number (iteration count, default 1024)
certs    -c file (certificate file) -k url (private-key URL)
racerts  -R file (RA certificate file) -k url (RA private-key URL)
keypair  -P url (private-key URL)
id        -I number (polling ID)
template -T file (certificate template)
          -s subject-ldap[;type=value]
          -u key-usage-name[;key-usage-name]
          -U extended-key-usage-name[;extended-key-usage-name]
access   URL where the CA listens for requests.
name     LDAP name for the issuing CA (if -C is not given).
```

Key URLs are either valid external key paths or in the format:

```
"generate://savetype:passphrase@keytype:size/save-file-prefix"
"file://passphrase/relative-key-file-path"
"file:relative-key-file-path"
"any-key-file-path"
```

The key generation "savetype" can be:

- ssh2, secsh2, secsh (Secure Shell 2 key type)
- ssh1, secsh1 (legacy Secure Shell 1 key type)
- pkcs1 (PKCS #1 format)
- pkcs8s (passphrase-protected PKCS #8, "shrouded PKCS #8")

```
- pkcs8 (plain-text PKCS #8)
- x509 (SSH-proprietary X.509 library key type)

-h Prints usage message.
-F Prints key usage extension and keytype instructions.
-e Prints command-line examples.
```

## Description

The **ssh-cmpclient-g3** command-line tool (**ssh-cmpclient-g3.exe** on Windows) is a certificate enrollment client that uses the CMP protocol. It can generate an RSA or DSA public-key pair and get certificates for their public components. CMP is specified by the IETF PKIX Working Group for certificate life-cycle management, and is supported by some CA platforms, such as Entrust PKI and RSA Keon.

## Commands

The **ssh-cmpclient-g3** command-line command keywords are listed below. Shorthands longer than three letters can be used to identify the command. The commands are case-insensitive. The user must specify the CA address URL for each command. Here the term "user" refers to a user, program, or hardware device.

### INITIALIZE

Requests the user's initial certificate. The request is authenticated using the reference number and the corresponding key (PSK) received from the CA or RA using some out-of-band mechanism.

The user must specify the PSK, the asymmetric key pair, and a subject name.

### ENROLL

Requests a new certificate when the user already has a valid certificate for the key. This request is similar to `initialize` except that it is authenticated using public-key methods.

### POLL

Polls for a certificate when a request was not immediately accepted.

### UPDATE

Requests an update of an existing certificate (replacement). The issued certificate will be similar to the existing certificate (names, flags, and other extensions). The user can change the key, and the validity times are updated by the CA. This request is authenticated by a valid existing key pair and a certificate.

### RECOVER

Requests recovery of a backed-up key. This request is authenticated either by PSK-based or certificate-based authentication. The template describes the certificate whose private key has already been backed up and should be recovered. Users can only recover keys they have backed up themselves.

### REVOKE

Requests revocation for a key specified in the template. Authentication of the request is made using a PSK or a certificate belonging to the same user as the subject of revocation.

**TUNNEL**

Operates in RA tunnel mode. Reads requests and optionally modifies the subject name, alternative names, and extensions based on the command line. Approves the request and sends it to the CA.

## Options

The **ssh-cmpclient-g3** command-line options are listed below. Note that when a file name is specified, an existing file with the same name will be overwritten. When subject names or other strings that contain spaces are given on the command line, they should be enclosed in double quotes.

**-B**

Requests private key backup to be performed for the initialize, enroll, and update commands.

**-o** *prefix*

Saves resulting certificates and CRLs into files with the given prefix. The prefix is first appended by a number, followed by the file extension `.crt` or `.crl`, depending on the type of object.

**-O** *filename*

Saves the result into the specified absolute filename. If there is more than one result file, the remaining results are rejected.

**-C** *file*

Specifies the file path that contains the CA certificate. If key backup is done, the file name must be given, but in most cases the LDAP name of the CA can be given instead.

**-S** *url*

Specifies the SOCKS URL if the CA is located behind a SOCKS-enabled firewall. The format of the URL is: `socks://[username@]server[:port][,/network/bits[,network/bits]]`

**-H** *url*

Uses the given HTTP proxy server to access the CA. The format of the URL is: `http://server[:port]/`

**-E**

Performs encryption proof of possession if the CA supports it. In this method of PoP, the request is not signed, but instead the PoP is established based on the ability to decrypt the certificates received from the CA. The CA encrypts the certificates with the user's public key before sending them to the user.

**-v** *num*

Selects the CMP protocol version. This is either value 1, for an RFC 2510-based protocol, or 2 (the default) for CMPv2.

**-N** *file*

Specifies a file to be used as an entropy source during key generation.

The usage line uses the following meta commands:

**psk**

The reference number and the corresponding key value given by the CA or RA.

**-p** *refnum:key/file*

*refnum* and *key* are character strings shared among the CA and the user. *refnum* identifies the secret key used to authenticate the message. The *refnum* string must not contain colon characters.

Alternatively, a filename containing the reference number and the key can be given as the argument.

**-i** *number*

*number* indicates the key hashing iteration count.

**certs**

The user's existing key and certificate for authentication.

**-k** *url*

URL specifying the private key location. This is an external key URL whose format is specified in [the section called “Synopsis”](#).

**-c** *file*

Path to the file that contains the certificate issued to the public key given in the **-k** option argument.

**racerts**

In RA mode, the RA key and certificate for authentication.

**-k** *url*

URL specifying the private key location. This is an external key URL whose format is specified in [the section called “Synopsis”](#).

**-R** *file*

Path to the file that contains the RA certificate issued to the public key given in the **-k** option argument.

**keypair**

The subject key pair to be certified.

**-P** *url*

URL specifying the private key location. This is an external key URL whose format is specified in [the section called “Synopsis”](#).

**id**

Polling ID used if the PKI action is left pending.

**-I** *number*

Polling transaction ID *number* given by the RA or CA if the action is left pending.

**template**

The subject name and flags to be certified.



`-T file`

The file containing the certificate used as the template for the operation. Values used to identify the subject are read from this, but the user can overwrite the key, key-usage flags, or subject names.

`-s subject-ldap[;type=value]*`

A subject name in reverse LDAP format, that is, the most general component first, and alternative subject names. The name `subject-ldap` will be copied into the request verbatim.

A typical choice would be a DN in the format "C=US,O=SSH,CN=Some Body", but in principle this can be anything that is usable for the resulting certificate.

The possible `type` values are `ip`, `email`, `dn`, `dns`, `uri`, and `rid`.

`-u key-usage-name[;key-usage-name]*`

Requested key usage purpose code. The following codes are recognized: `digitalSignature`, `non-Repudiation`, `keyEncipherment`, `dataEncipherment`, `keyAgreement`, `keyCertSign`, `cRLSign`, `encipherOnly`, `decipherOnly`, and `help`. The special keyword `help` lists the supported key usages which are defined in RFC 3280.

`-U extended-key-usage-name[;extended-key-usage-name]*`

Requested extended key usage code. The following codes, in addition to user-specified dotted OID values are recognized: `serverAuth`, `clientAuth`, `codeSigning`, `emailProtection`, `timeStamping`, `ikeIntermediate`, and `smartCardLogon`.

`access`

Specifies the CA address in URL format. Possible access methods are HTTP (`http://host:port/path`), or plain TCP (`tcp://host:port/path`). If the host address is an IPv6 address, it must be enclosed in square brackets (`http://[IPv6-address]:port/`).

`name`

Optionally specifies the destination CA name for the operation, in case a CA certificate was not given using the option `-c`.

## Examples

### Initial Certificate Enrollment

This example provides commands for enrolling an initial certificate for digital signature use. It generates a private key into a PKCS #8 plaintext file named `initial.prv`, and stores the enrolled certificate into file `initial-0.crt`. The user is authenticated to the CA with the key identifier (refnum) 62154 and the key `ssh`. The subject name and alternative IP address are given, as well as key-usage flags. The CA address is `pki.ssh.com`, the port 8080, and the CA name to access `Test CA 1`.

```
$ ssh-cmpclient-g3 INITIALIZE \
-P generate://pkcs8@rsa:1024/initial -o initial \
-p 62154:ssh \
```

```
-s 'C=FI,O=SSH,CN=Example/initial;IP=1.2.3.4' \
-u digitalsignature \
http://pki.ssh.com:8080/pkix/ \
'C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities'
```

As a response the command presents the issued certificate to the user, and the user accepts it by typing *yes* at the prompt.

```
Certificate =
  SubjectName = <C=FI, O=SSH, CN=Example/initial>
  IssuerName = <C=FI, O=SSH Communications Security Corp,
    CN=SSH Test CA 1 No Liabilities>
  SerialNumber= 8017690
  SignatureAlgorithm = rsa-pkcs1-sha1
  Validity = ...
  PublicKeyInfo = ...
  Extensions =
    Viewing specific name types = IP = 1.2.3.4
    KeyUsage = DigitalSignature
    CRLDistributionPoints = ...
    AuthorityKeyID =
      KeyID = 3d:cb:be:20:64:49:16:1d:88:b7:98:67:93:f0:5d:42:81:2e:bd:0c
    SubjectKeyID =
      KeyID = 6c:f4:0e:ba:b9:ef:44:37:db:ad:1f:fc:46:e0:25:9f:c8:ce:cb:da
  Fingerprints =
    MD5 = b7:6d:5b:4d:e0:94:d1:1f:ec:ca:c2:ed:68:ac:bf:56
    SHA-1 = 4f:de:73:db:ff:e8:7d:42:c4:7d:e1:79:1f:20:43:71:2f:81:ff:fa

Do you accept the certificate above? yes
```

## Key update

Before the certificate expires, a new certificate with updated validity period should be enrolled. **ssh-cmpclient-g3** supports key update, where a new private key is generated and the key update request is authenticated with the old (still valid) certificate. The old certificate is also used as a template for issuing the new certificate, so the identity of the user will not be changed during the key update. With the following command you can update the key pair, which was enrolled in the previous example. Presenting the resulting certificate has been left out.

```
$ ssh-cmpclient-g3 UPDATE \
-k initial.prv -c initial-0.crt -P \
generate://pkcs8@rsa:1024/updatedcert -o updatedcert \
http://pki.ssh.com:8080/pkix/ \
"C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities"
```

The new key pair can be found in the files with the `updatedcert` prefix. The policy of the issuing CA needs to also allow automatic key updates if **ssh-cmpclient-g3** is used in the `UPDATE` mode.

## ssh-scepclient-g3

ssh-scepclient-g3 -- SCEP enrollment client

### Synopsis

ssh-scepclient-g3 command [options] access [name]

Where command is one of the following:

```
GET-CA
GET-CHAIN
ENROLL psk keypair template
```

Most commands can accept the following options:

```
-o prefix      Save result into files with prefix.
-S url        Use this socks server to access CA.
-H url        Use this HTTP proxy to access CA.
```

The following identifiers are used to specify options:

```
psk      -p key (used as revocationPassword or challengePassword)
keypair  -P url (private-key URL)
ca        -C file (CA certificate file)
          -E file (RA encryption certificate file)
          -V file (RA validation certificate file)
template -T file (certificate template)
          -s subject-ldap[;type=value]
          -u key-usage-name[;key-usage-name]
          -U extended-key-usage-name[;extended-key-usage-name]
access   URL where the CA listens for requests.
```

GET-CA and GET-CHAIN take name argument, that is something interpreted by the CA to specify a CA entity managed by the responder.

Key URLs are either valid external key paths or in the format:

```
"generate://savetype:password@keytype:size/save-file-prefix"
"file://savetype:password@/file-prefix"
"file://passphrase/file-prefix"
"file:/file-prefix"
"key-filename"
```

The "keytype" for the SCEP protocol has to be "rsa".

The key generation "savetype" can be:

- ssh2 (Secure Shell 2 key type)
- ssh1 (Legacy Secure Shell 1 key type)
- ssh (SSH proprietary crypto library format, passphrase-protected)
- pkcs1 (PKCS#1 format)
- pkcs8s (passphrase-protected PKCS#8, "shrouded PKCS#8")
- pkcs8 (plain-text PKCS#8)
- x509 (SSH proprietary X.509 library key type)

## Description

The **ssh-scepclient-g3** command-line tool (**ssh-scepclient-g3.exe** on Windows) is a certificate enrollment client that uses the SCEP protocol. It can generate an RSA public-key pair and get certificates for its public components. The SCEP protocol was developed by Cisco and Verisign to be used on Cisco routers. Nowadays most CA platforms support this protocol for client certificate enrollment.

## Commands

The **ssh-scepclient-g3** command-line command keywords are listed below. Shorthands longer than three letters can be used to identify the command. The commands are case-insensitive. The user must specify the CA address URL for each command. Here the term "user" refers to a user, program, or hardware device.

### GET-CA

Requests CA or RA certificate download from the CA, and display the certificate fingerprint for CA validation. Fingerprints should be received from the CA using some out-of-band mechanism.

### GET-CHAIN

Requests certificate chain from the CA/RA to the top-level CA.

### ENROLL

Requests a new certificate from the CA. The CA will authorize the request using some out-of-band mechanism, or it can contain a password received from the CA.

## Options

### -o *prefix*

Saves output certificates into files with the given prefix. The prefix is first appended by a number, followed by the file extension `.ca` for CA certificates or `.cert` for user certificates.

### -S *url*

Specifies the SOCKS URL if the CA is located behind a SOCKS-enabled firewall. The format of the URL is: `socks://[username@]server[:port][ /network/bits[,network/bits]]`

### -H *url*

Uses the given HTTP proxy server to access the CA. The format of the URL is: `http://server[:port]/.`

The usage line uses the following meta commands:

### psk

The pre-shared key given by the CA or RA, or a revocation password invented by the client and provided to the CA when the user wishes to revoke the certificate issued. The type and need for this depends on the PKI platform used by the CA.

`-p key`

An authentication password or a revocation password transferred (in encrypted format) to the CA for certification request or revocation request authorization purposes.

`keypair`

The subject key pair to be certified.

`-P url`

URL specifying the private key location. This is an external key URL whose format is specified in [the section called “Synopsis”](#).

`ca`

The CA/RA certificates.

`-C file`

When performing enrollment, reads the CA certificate from the given file path.

`-E file`

Optionally specifies the RA encryption certificate.

`-V file`

Optionally specifies the RA signing certificate.

`template`

The subject name and flags to be certified.

`-T file`

The file containing the certificate used as the template for the operation. Values used to identify the subject are read from this, but the user may overwrite the key, key-usage flags, or subject names.

`-s subject-ldap[;type=value]*`

A subject name in reverse LDAP format, that is, the most general component first, and alternative subject names. The name `subject-ldap` will be copied into the request verbatim.

A typical choice would be a DN in the format `"C=US,O=SSH,CN=Some Body"`, but in principle this can be anything that is usable for the resulting certificate.

The possible `type` values are `ip`, `email`, `dn`, `dns`, `uri`, and `rid`.

`-u key-usage-name[;key-usage-name]*`

Requested key usage purpose code. The following codes are recognized: `digitalSignature`, `non-Repudiation`, `keyEncipherment`, `dataEncipherment`, `keyAgreement`, `keyCertSign`, `cRLSign`, `encipherOnly`, `decipherOnly`, and `help`. The special keyword `help` lists the supported key usages which are defined in *RFC 3280*.

`-U extended-key-usage-name[;extended-key-usage-name]*`

Requested extended key usage code. The following codes, in addition to user-specified dotted OID values are recognized: `serverAuth`, `clientAuth`, `codeSigning`, `emailProtection`, `timeStamping`, `ikeIntermediate`, and `smartCardLogon`.

`access`

Specifies the address of the CA in URL format. If the host address is an IPv6 address, it must be enclosed in brackets (`http://[IPv6-address]:port/`).

`name`

Specifies the destination CA name.

## Examples

In the following example we first receive the CA certificate. The CA address is `pki.ssh.com`, the port is 8080, and the CA name is `test-cal.ssh.com`.

```
$ ssh-scepclient-g3 GET-CA \
  -o ca http://pki.ssh.com:8080/scep/ \
  test-cal.ssh.com

Received CA/RA certificate ca-0.ca:

fingerprint 9b:96:51:bb:29:0d:c9:e0:75:c8:03:0d:0d:92:60:6c
```

Next, we enroll an RSA certificate. The user is authenticated to the CA with the key `ssh`. The subject name and alternative IP address are given, as well as key-usage flags.

```
$ ssh-scepclient-g3 ENROLL \
  -C ca-0.ca -p ssh \
  -o subject -P generate://pkcs8:ssh@rsa:1024/subject \
  -s 'C=FI,O=SSH,CN=SCEP Example;IP=1.2.3.4' \
  -u digitalsignature \
  http://pki.ssh.com:8080/scep/

Received user certificate subject-0.crt:
fingerprint 4b:7e:d7:67:27:5e:e0:54:2f:5b:56:69:b5:01:d2:15
$ ls subject*
subject-0.crt  subject.prv
```

## ssh-certview-g3

`ssh-certview-g3 -- certificate viewer`

## Synopsis

`ssh-certview-g3`

[options...] *file*  
[options...] *file* ...

## Description

The **ssh-certview-g3** program (**ssh-certview-g3.exe** on Windows) is a simple command-line application, capable of decoding and showing X.509 certificates, CRLs, and certification requests. The command output is written to the standard output.

## Options

The following options are available:

- h  
Displays a short help.
- verbose  
Gives more diagnostic output.
- quiet  
Gives no diagnostic output.
- auto  
The next input file type is auto-detected (default).
- cert  
The next input file is a certificate.
- certpair  
The next input file is a cross-certificate pair.
- crmf  
The next input file is a CRMF certification request.
- req  
The next input file is a PKCS #10 certification request.
- crl  
The next input file is a CRL.
- prv  
The next input file is a private key.
- pkcs12  
The next input file is a PKCS#12 package.

`-ssh2`

The next input file is an SSH2 public key.

`-spkac`

The next input file is a Netscape-generated SPKAC request.

`-noverify`

Does not check the validity of the signature on the input certificate.

`-autoenc`

Determines PEM/DER automatically (default).

`-pem`

Assumes that the input file is in PEM (ASCII base-64) format. This option allows both actual PEM (with headers and footers), and plain base-64 (without headers and footers). An example of PEM header and footer is shown below:

```
-----BEGIN CERTIFICATE-----  
encoded data  
-----END CERTIFICATE-----
```

`-der`

Assumes that the input file is in DER format.

`-hexl`

Assumes that the input file is in Hexl format. (Hexl is a common Unix tool for outputting binary files in a certain hexadecimal representation.)

`-skip number`

Skips *number* bytes from the beginning of input before trying to decode. This is useful if the file contains some garbage before the actual contents.

`-ldap`

Prints names in LDAP order.

`-utf8`

Prints names in UTF-8.

`-latin1`

Prints names in ISO-8859-1.

`-base10`

Outputs big numbers in base-10 (default).

`-base16`

Outputs big numbers in base-16.



`-base64`

Outputs big numbers in base-64.

`-width number`

Sets output width (*number* characters).

## Example

For example, using a certificate downloaded from `pki.ssh.com`, when the following command is given:

```
$ ssh-certview-g3 -width 70 ca-certificate.cer
```

The following output is produced:

```
Certificate =
  SubjectName = <C=FI, O=SSH Communications Security Corp, CN=Secure
    Shell Test CA>
  IssuerName = <C=FI, O=SSH Communications Security Corp, CN=Secure
    Shell Test CA>
  SerialNumber= 34679408
  SignatureAlgorithm = rsa-pkcs1-sha1
  Certificate seems to be self-signed.
    * Signature verification success.
  Validity =
    NotBefore = 2003 Dec 3rd, 08:04:27 GMT
    NotAfter = 2005 Dec 2nd, 08:04:27 GMT
  PublicKeyInfo =
    PublicKey =
      Algorithm name (SSH) : if-modn{sign{rsa-pkcs1-md5}}
      Modulus n (1024 bits) :
        9635680922805930263476549641957998756341022541202937865240553
        9374740946079473767424224071470837728840839320521621518323377
        3593102350415987252300817926769968881159896955490274368606664
        0759644131690750532665266218696466060377799358036735475902257
        6086098562919363963470926690162744258451983124575595926849551
        903
      Exponent e ( 17 bits) :
        65537
    Extensions =
      Available = authority key identifier, subject key identifier, key
        usage(critical), basic constraints(critical), authority
        information access
      KeyUsage = DigitalSignature KeyEncipherment KeyCertSign CRLSign
        [CRITICAL]
      BasicConstraints =
        PathLength = 0
        cA = TRUE
        [CRITICAL]
      AuthorityKeyID =
        KeyID =
```

```
    eb:f0:4d:b5:b2:4c:be:47:35:53:a8:37:d2:8d:c8:b2:f1:19:71:79
SubjectKeyID =
  KeyId =
    eb:f0:4d:b5:b2:4c:be:47:35:53:a8:37:d2:8d:c8:b2:f1:19:71:79
AuthorityInfoAccess =
  AccessMethod = 1.3.6.1.5.5.7.48.1
  AccessLocation =
    Following names detected =
      URI (uniform resource indicator)
    Viewing specific name types =
      URI = http://pki.ssh.com:8090/ocsp-1/
Fingerprints =
  MD5 = c7:af:e5:3d:f6:ea:ce:da:07:93:d0:06:8d:c0:0a:f8
  SHA-1 =
    27:d7:19:47:7c:08:3e:1a:27:4b:68:8e:18:83:e8:f9:23:e8:29:85
```

## ssh-ekview-g3

ssh-ekview-g3 -- external key viewer

### Synopsis

ssh-ekview-g3 [options...] *provider*

### Description

The **ssh-ekview-g3** program (**ssh-ekview-g3.exe** on Windows) allows you to export certificates from external key providers such as Entrust. You can further study these certificates with **ssh-certview-g3**.

This is useful when you want to generate, for example, entries for allowing certificate authentication in the `ssh-server-config.xml` file. You might need to know the subject names on the certificate.

With **ssh-ekview-g3**, you can export the certificate and get the information you need from the certificates with **ssh-certview-g3**.

### Options

The following options are available:

-h

Displays a short help.

-i *info*

Uses *info* as the initialization string for the provider.

-k

Prints the key paths only.

-e *keypath*

Exports certificates at *keypath* to files.

-a

Exports all found certificates to files.

-b *base*

Uses *base* when printing integers. For example, the decimal 10 is 'a' in base-16.

## Example

For example the following command will dump all certificates in the `entrust` provider to files:

```
ssh-ekview-g3 -a -i"ini-file($HOME/my.ini) profile-file($HOME/solo.ini)" entrust
```



# Appendix D Egrep Syntax

The SSH Tectia tunneling filter rules can be matched to hostname or IP address patterns specified using the **egrep** syntax. In addition, regular expressions can be used in selectors when specifying ranges of values. The **egrep** syntax is explained in this section.

## D.1 Egrep Patterns

The escape character is a backslash (\). You can use it to escape meta characters to use them in their plain character form.

In the following examples literal 'E' and 'F' denote any expression, whether a pattern or a character.

(  
    Start a capturing subexpression.

)  
    End a capturing subexpression.

E|F  
    Disjunction, match either E or F (inclusive). E is preferred if both match.

E\*  
    Act as Kleene star, match E zero or more times.

E+  
    Closure, match E one or more times.

E?  
    Option, match E optionally once.

.  
    Match any character except for newline characters (\n, \f, \r) and the NULL byte.

E{n}  
    Match E exactly n times.

$E\{n,\}$  or  $E\{n,0\}$

Match  $E$   $n$  or more times.

$E\{,n\}$  or  $E\{0,n\}$

Match  $E$  at most  $n$  times.

$E\{n,m\}$

Match  $E$  no less than  $n$  times and no more than  $m$  times.

[

Start a character set, see [Section D.3](#).

\$

Match the empty string at the end of the input or at the end of a line.

^

Match the empty string at the start of the input or at the beginning of a line.

## D.2 Escaped Tokens for Regex Syntax Egrep

$\backslash 0n..n$

The literal byte with octal value  $n..n$ .

$\backslash 0$

The `NULL` byte.

$\backslash [1-9]..x$

The literal byte with decimal value  $[1-9]..x$ .

$\backslash xn..n$  or  $\backslash 0xn..n$

The literal byte with hexadecimal value  $n..n$ .

$\backslash <$

Match the empty string at the beginning of a word.

$\backslash >$

Match the empty string at the end of a word.

$\backslash b$

Match the empty string at a word boundary.

$\backslash B$

Match the empty string provided it is not at a word boundary.

$\backslash w$

Match a word-constituent character, equivalent to  $[a-zA-Z0:9-]$ .

`\W`

Match a non-word-constituent character.

`\a`

Literal alarm character.

`\e`

Literal escape character.

`\f`

Literal line feed.

`\n`

Literal new line, equivalent to C's `\n` so it can be more than one character long.

`\r`

Literal carriage return.

`\t`

Literal tab.

All other escaped characters denote the literal character itself.

## D.3 Character Sets For Egrep

A character set starts with '[' and ends at non-escaped ']' that is not part of a POSIX character set specifier and that does not follow immediately after '['.

The following characters have a special meaning and need to be escaped if meant literally:

- (minus sign)

A range operator, except immediately after '[', where it loses its special meaning.

^

If immediately after the starting '[', denotes a complement: the whole character set will be complemented. Otherwise literal '^'.

`[:alnum:]`

Characters for which 'isalnum' returns true.

`[:alpha:]`

Characters for which 'isalpha' returns true.

`[:cntrl:]`

Characters for which 'iscntrl' returns true.

`[:digit:]`

Characters for which 'isdigit' returns true.

`[:graph:]`

Characters for which 'isgraph' returns true.

`[:lower:]`

Characters for which 'islower' returns true.

`[:print:]`

Characters for which 'isprint' returns true.

`[:punct:]`

Characters for which 'ispunct' returns true.

`[:space:]`

Characters for which 'isspace' returns true.

`[:upper:]`

Characters for which 'isupper' returns true.

`[:xdigit:]`

Characters for which 'isxdigit' returns true.

**Example:** `[[:xdigit:]]XY` is typically equivalent to `[0123456789ABCDEFabcdefXY]`.

It is also possible to include the predefined escaped character sets into a newly defined one, so `[\d\s]` matches digits and whitespace characters.

Also, escape sequences resulting in literals work inside character sets.



## Appendix E Audit Messages

This appendix lists the audit messages generated by the Connection Broker.

### **1000 KEX\_failure**

**Level:** warning

**Origin:** SSH Tectia Server, Connection Broker

The key exchange failed.

Default log facility: normal

Argument	Description
Username	User's login name (not present for first KEX)
Algorithm	KEX algorithm name (not present if failure happens before choosing the algorithm)
Text	Error description
Session-Id	Session identifier

### **1001 Algorithm\_negotiation\_failure**

**Level:** warning

**Origin:** SSH Tectia Server, Connection Broker

Algorithm negotiation failed - there was no common algorithm in the client's and server's lists.

Default log facility: normal

Argument	Description
Username	User's login name (not present for first KEX)
Algorithm	Algorithm type
Client algorithms	Client's algorithm list
Server algorithms	Server's algorithm list
Session-Id	Session identifier

### **1002 Algorithm\_negotiation\_success**

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

Algorithm negotiation succeeded.

Default log facility: normal

Argument	Description
----------	-------------

Username	User's login name (not present for first KEX)
----------	---

Text	Negotiated algorithms
------	-----------------------

Session-Id	Session identifier
------------	--------------------

### 1003 KEX\_success

**Level:** informational

**Origin:** Connection Broker

Key-exchange was successful.

Default log facility: normal

Argument	Description
----------	-------------

Session-Id	Session identifier.
------------	---------------------

Protocol-session-Id	Protocol session identifier.
---------------------	------------------------------

### 1100 Certificate\_validation\_failure

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

A received certificate failed to validate correctly under any of the configured CAs.

Default log facility: normal

Argument	Description
----------	-------------

Username	User's login name (not present for first KEX)
----------	---

Text	Resulting search states for all configured CAs.
------	---

Session-Id	Session identifier
------------	--------------------

### 1101 Certificate\_validation\_success

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

A received certificate validated correctly under one or more configured CAs.

Default log facility: normal

Argument	Description
----------	-------------

Username	User's login name
----------	-------------------

CA List	A list of CAs under which the user's certificate validated correctly.
---------	---

Session-Id	Session identifier
------------	--------------------

**1110 CM\_find\_started****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

A low-level search was started in the certificate validation subsystem.

Default log facility: normal

Argument	Description
Ctx	Search context
Search constraints	Search constraints.

**1111 CM\_find\_finished****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

A low-level find operation has finished in the certificate validation subsystem.

Default log facility: normal

Argument	Description
Ctx	Context pointer that identifies the search

**1112 CM\_cert\_not\_in\_search\_interval****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

The certificate is not valid during the required time period.

Default log facility: normal

Argument	Description
SubjectName	Subject name of the certificate
Text	Error description
Ctx	Search context

**1113 CM\_certificate\_revoked****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

A certificate was found to be revoked.

Default log facility: normal

Argument	Description
SubjectName	Subject name of the certificate
Ctx	The context pointer of the search

**1114 CM\_cert\_search\_constraint\_mismatch****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

The certificate did not satisfy the constraints set for the search.

Default log facility: normal

Argument	Description
SubjectName	Subject name of the certificate
Text	Description of the mismatch
Ctx	Search context

**1115 CM\_ldap\_search\_started****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

An LDAP search for a CRL or a sub-CA is being started.

Default log facility: normal

Argument	Description
Text	Search details

**1116 CM\_ldap\_search\_success****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

An LDAP search for a CRL or a sub-CA completed successfully.

Default log facility: normal

Argument	Description
Text	Search details

**1117 CM\_ldap\_search\_failure****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

The attempt to contact an LDAP server was unsuccessful.

Default log facility: normal

Argument	Description
Text	Error details

**1118 CM\_http\_search\_started****Level:** informational

---

**Origin:** SSH Tectia Server, Connection Broker

The certificate validation subsystem is initiating a search for a CRL or a sub-CA through the HTTP protocol.

Default log facility: normal

Argument	Description
Text	Search target

#### **1119 CM\_http\_search\_success**

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

An HTTP request for a CRL or a sub-CA completed successfully.

Default log facility: normal

Argument	Description
Text	Status message detailing what was being retrieved

#### **1120 CM\_http\_search\_failure**

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

An HTTP request for a CRL or a sub-CA failed.

Default log facility: normal

Argument	Description
Text	Error details

#### **1121 CM\_crl\_added**

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

A new CRL was successfully added to the certificate validation subsystem.

Default log facility: normal

Argument	Description
Text	CRL's issuer and validity period

#### **1122 Certificate\_end\_point\_id\_check\_success**

**Level:** informational

**Origin:** Connection Broker

End point identity check succeeded.

Default log facility: normal

Argument	Description
Server	Host name
Text	Explanatory message

**1123 Certificate\_end\_point\_id\_check\_warning****Level:** informational**Origin:** Connection Broker

Certificate end point identity check warning.

Default log facility: normal

Argument	Description
Server	Host name
Text	Warning message

**1124 Certificate\_end\_point\_id\_check\_failure****Level:** informational**Origin:** Connection Broker

Certificate end point identity check failure.

Default log facility: normal

Argument	Description
Server	Host name
Text	Error message

**1200 Key\_store\_create****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

Key store created.

Default log facility: normal

**1201 Key\_store\_create\_failed****Level:** warning**Origin:** SSH Tectia Server, Connection Broker

Key store creation failed.

Default log facility: normal

**1202 Key\_store\_destroy****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

Key store destroyed.

Default log facility: normal

#### **1204 Key\_store\_add\_provider**

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

Added a provider to the key store.

Default log facility: normal

<b>Argument</b>	<b>Description</b>
Type	Provider type

#### **1205 Key\_store\_add\_provider\_failed**

**Level:** warning

**Origin:** SSH Tectia Server, Connection Broker

Adding a provider to the key store failed.

Default log facility: normal

<b>Argument</b>	<b>Description</b>
Type	Provider type
EK error	Error message

#### **1206 Key\_store\_remove\_provider**

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

Removed a provider from the key store.

Default log facility: normal

<b>Argument</b>	<b>Description</b>
Init info	Provider name

#### **1208 Key\_store\_decrypt**

**Level:** informational

**Origin:** SSH Tectia Server, Connection Broker

A key was used successfully for decryption.

Default log facility: normal

<b>Argument</b>	<b>Description</b>
Key path	Key path

Argument	Description
Fwd path	Fwd path

**1209 Key\_store\_decrypt\_failed****Level:** warning**Origin:** SSH Tectia Server, Connection Broker

A key was used unsuccessfully for decryption.

Default log facility: normal

Argument	Description
Key path	Key path
Fwd path	Fwd path
Crypto error	Error string

**1210 Key\_store\_sign****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

A key was used successfully for signing.

Default log facility: normal

Argument	Description
Key path	Key path
Fwd path	Fwd path

**1211 Key\_store\_sign\_failed****Level:** warning**Origin:** SSH Tectia Server, Connection Broker

A key was used unsuccessfully for signing.

Default log facility: normal

Argument	Description
Key path	Key path
Fwd path	Fwd path
Crypto error	Error string

**1212 Key\_store\_sign\_digest****Level:** informational**Origin:** SSH Tectia Server, Connection Broker

A key was used successfully for signing a digest.

Default log facility: normal



Argument	Description
Key path	Key path
Fwd path	Fwd path

### 1213 Key\_store\_sign\_digest\_failed

**Level:** warning

**Origin:** SSH Tectia Server, Connection Broker

A key was used unsuccessfully for signing a digest.

Default log facility: normal

Argument	Description
Key path	Key path
Fwd path	Fwd path
Crypto error	Error string

### 1214 Key\_store\_ek\_provider\_failure

**Level:** warning

**Origin:** SSH Tectia Server, Connection Broker

External key provider failure.

Default log facility: normal

Argument	Description
Key path	Key path
Text	Key label

### 1300 Channel inbound statistics

**Level:** informational

**Origin:** Connection Broker, SSH Tectia Server

Statistics for the inbound side of a channel (traffic arriving from the network)

Default log facility: normal

Argument	Description
Username	User's login name
Session-Id	Session identifier
Channel Id	Local channel id
Packet count	Protocol packet count
Packet size	Average protocol packet payload size

### 1301 Channel outbound statistics

**Level:** informational

**Origin:** Connection Broker, SSH Tectia Server

Statistics for the outbound side of a channel (traffic going to the network)

Default log facility: normal

Argument	Description
Username	User's login name
Session-Id	Session identifier
Channel Id	Local channel id
Packet count	Protocol packet count
Packet size	Average protocol packet payload size
Packet size	Final size of outbound channel buffer

#### **6000 Broker\_client\_connect**

**Level:** informational

**Origin:** Connection Broker

A client connected to the Broker.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Process id
Local username	Local user name

#### **6001 Broker\_client\_connect\_failed**

**Level:** warning

**Origin:** Connection Broker

A client attempted to connect unsuccessfully to the Broker.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Process id
Local username	Local user name
Text	Reason

#### **6002 Broker\_client\_disconnect**

**Level:** informational

**Origin:** Connection Broker

A client disconnected from the Broker.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Process id
Local username	Local user name

#### **6004 Broker\_exec\_channel\_open**

**Level:** informational

**Origin:** Connection Broker

The Broker opened an exec channel.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Command	Command
Text	Exec parameters
Channel Id	Channel ID
Session-Id	Session ID

#### **6005 Broker\_exec\_channel\_open\_failed**

**Level:** warning

**Origin:** Connection Broker

The Broker failed to open an exec channel for a client.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Command	Command
Text	Exec parameters
Channel Id	Channel ID
Text	Reason
Session-Id	Session ID

**6006 Broker\_tunnel\_open****Level:** informational**Origin:** Connection Broker

The Broker opened a tunnel for a client.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Dst	Destination host
Dst Port	Destination port
Tunnel type	Tunnel type
Session-Id	Session ID

**6007 Broker\_tunnel\_open\_failed****Level:** warning**Origin:** Connection Broker

The Broker failed to open a tunnel for a client.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Dst	Destination host
Dst Port	Destination port
Tunnel type	Tunnel type
Text	Reason
Session-Id	Session ID

**6008 Broker\_tunnel\_listener\_open****Level:** informational**Origin:** Connection Broker

The Broker opened a tunnel listener for a client.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Listener	Listener host
Listener Port	Listener port
Dst	Destination host
Dst Port	Destination port
Tunnel type	Tunnel type
Text	Tunnel listener parameters
Session-Id	Session ID

#### **6009 Broker\_tunnel\_listener\_open\_failed**

**Level:** warning

**Origin:** Connection Broker

The Broker failed to open a tunnel listener for a client.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Listener	Listener host
Listener Port	Listener port
Dst	Destination host
Dst Port	Destination port
Tunnel type	Tunnel type
Text	Tunnel listener parameters
Text	Reason
Session-Id	Session ID

#### **6010 Broker\_channel\_fd\_strip**

**Level:** informational

**Origin:** Connection Broker

The Broker destroyed a channel object (and returned the underlying fd to the client).

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Text	Channel permanent?
Local username	Local user name
Session-Id	Session ID

#### **6011 Broker\_channel\_fd\_strip\_failed**

**Level:** warning

**Origin:** Connection Broker

The Broker failed to destroy a channel object (and return the underlying fd to the client).

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Text	Channel permanent?
Local username	Local user name
Text	Reason
Session-Id	Session ID

#### **6012 Broker\_channel\_control**

**Level:** informational

**Origin:** Connection Broker

The Broker sent a channel control message.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Command	Command
Args	Arguments
Local username	Local user name
Session-Id	Session ID

**6013 Broker\_channel\_control\_failed****Level:** warning**Origin:** Connection Broker

The Broker failed to send a channel control message.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Command	Command
Args	Arguments
Local username	Local user name
Text	Reason
Session-Id	Session ID

**6014 Broker\_channel\_close****Level:** informational**Origin:** Connection Broker

The Broker closed a channel.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Exit Value	Exit value
Local username	Local user name
Session-Id	Session ID

**6015 Broker\_channel\_close\_failed****Level:** warning**Origin:** Connection Broker

The Broker failed to close a channel.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID

Argument	Description
Local username	Local user name
Text	Reason

**6018 Broker\_server\_version\_request****Level:** informational**Origin:** Connection Broker

The Broker requested (and got) the server version.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Ver	Version
Local username	Local user name
Session-Id	Session ID

**6019 Broker\_server\_version\_request\_failed****Level:** warning**Origin:** Connection Broker

The Broker failed to get the server version.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Local username	Local user name
Text	Reason
Session-Id	Session ID

**6020 Broker\_channel\_process\_exit****Level:** informational**Origin:** Connection Broker

Channel process exit request was successful.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process id



**Argument**

Local username

Session-Id

**Description**

Local user name

Session ID

**6021 Broker\_channel\_process\_exit\_failed****Level:** warning**Origin:** Connection Broker

Channel process exit request failed.

Default log facility: normal

**Argument**

Client

Pid

Text

Local username

Session-Id

**Description**

Client name

Client process id

Reason

Local user name

Session ID

**6025 Broker\_connector\_license\_check\_failed****Level:** warning**Origin:** Connection Broker

Connector license check failed.

Default log facility: normal

**Argument**

Text

Session-Id

**Description**

Error message

Session identifier

**6026 Broker\_server\_rekey****Level:** notice**Origin:** Connection Broker

The Broker requested rekeying and it was successful.

Default log facility: normal

**Argument**

Client

Pid

Channel Id

Local username

Session-Id

**Description**

Client name

Client process id

Channel ID

Local user name

Session ID

**6027 Broker\_server\_rekey\_failed**

**Level:** warning

**Origin:** Connection Broker

The Broker requested rekeying but it failed.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Channel Id	Channel ID
Local username	Local user name
Text	Reason
Session-Id	Session ID

### 6100 Broker\_starting

**Level:** notice

**Origin:** Connection Broker

The Broker is starting.

Default log facility: normal

Argument	Description
Local username	Local user name

### 6101 Broker\_start\_failed

**Level:** warning

**Origin:** Connection Broker

Starting the Broker failed.

Default log facility: normal

Argument	Description
Local username	Local user name
Success   Error	Error code
Text	Error message

### 6102 Broker\_running

**Level:** notice

**Origin:** Connection Broker

The Broker is running.

Default log facility: normal

**Argument**

Local username

Text

**Description**

Local user name

Message text

**6104 Broker\_stopping****Level:** notice**Origin:** Connection Broker

The Broker is stopping.

Default log facility: normal

**Argument**

Local username

**Description**

Local user name

**6106 Broker\_reconfig\_started****Level:** notice**Origin:** Connection Broker

Reconfiguration started.

Default log facility: normal

**Argument**

Local username

**Description**

Local user name

**6108 Broker\_reconfig\_finished****Level:** notice**Origin:** Connection Broker

Reconfiguration finished.

Default log facility: normal

**Argument**

Local username

Success | Error

**Description**

Local user name

Error code

**6114 Broker\_config\_deprecated\_element****Level:** warning**Origin:** Connection Broker

The Broker config contains a deprecated element.

Default log facility: normal

**Argument**

Text

**Description**

Event description.

**6200 Broker\_tcp\_connect****Level:** informational**Origin:** Connection Broker

Broker TCP connection attempt was successful.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Src Port	Source port
Local username	Local username

**6201 Broker\_tcp\_connect\_failed****Level:** warning**Origin:** Connection Broker

Broker TCP connection attempt failed.

Default log facility: normal

Argument	Description
Dst	Destination host
Dst Port	Destination port
Local username	Local username
NIO error	NIO error

**6204 Broker\_transport\_connect****Level:** informational**Origin:** Connection Broker

A transport was connected through TCP.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Remote username	Remote username
Src Port	Source port
Local username	Local username
Session-Id	Session ID

**6206 Broker\_transport\_gateway\_connect****Level:** informational**Origin:** Connection Broker

A transport was connected through a gateway handle.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Remote username	Remote username
Local username	Local username
Session-Id	Session ID

### **6208 Broker\_connection\_connect**

**Level:** informational

**Origin:** Connection Broker

The Broker got successfully a Secure Shell connection up.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Local username	Local username
Remote username	Remote username
Uses gateway?	Is this going through a gateway handle
Session-Id	Session ID

### **6209 Broker\_connection\_connect\_failed**

**Level:** warning

**Origin:** Connection Broker

The Broker failed to get a Secure Shell connection up.

Default log facility: normal

Argument	Description
Dst	Destination host
Dst Port	Destination port
Local username	Local username
Remote username	Remote username
Uses gateway?	Is this going through a gateway handle
Session-Id	Session ID
Text	Error code

### **6210 Broker\_connection\_disconnect**

**Level:** informational

**Origin:** Connection Broker

A Secure Shell connection initiated by the Broker was disconnected.

Default log facility: discard

Argument	Description
Local username	Local user
Session-Id	Session identifier

#### **6211 Broker\_unknown\_hostkey\_accepted**

**Level:** warning

**Origin:** Connection Broker

\* The Broker accepted an unknown hostkey without user interaction \* because of configuration.

Default log facility: normal

Argument	Description
Text	Key digest
Dst	Destination host
Dst Port	Destination port
Local username	Local username
Remote username	Remote username

#### **6212 Broker\_new\_hostkey**

**Level:** warning

**Origin:** Connection Broker

\* First connection to a server or this server hostkey was never \* saved before.

Default log facility: normal

Argument	Description
Text	Key digest
Dst	Destination host
Dst Port	Destination port
Local username	Local username
Remote username	Remote username

#### **6213 Broker\_hostkey\_changed**

**Level:** warning

**Origin:** Connection Broker

\* Server hostkey is different than the saved hostkey.

Default log facility: normal

**Argument**

Text

Dst

Dst Port

Local username

Remote username

**Description**

Key digest

Destination host

Destination port

Local username

Remote username

**6301 Broker\_userauth\_failure****Level:** warning**Origin:** Connection Broker

User authentication failed.

Default log facility: normal

**Argument**

Text

Session-Id

**Description**

Reason

Session identifier

**6302 Broker\_userauth\_method\_success****Level:** informational**Origin:** Connection Broker

A user authentication method succeeded.

Default log facility: discard

**Argument**

Text

Session-Id

**Description**

Authentication method

Session identifier

**6303 Broker\_userauth\_method\_failure****Level:** warning**Origin:** Connection Broker

A user authentication method failed.

Default log facility: discard

**Argument**

Text

Text

Session-Id

**Description**

Authentication method

Reason

Session identifier

**6401 Connector\_filter\_rule****Level:** informational**Origin:** Connection Broker

Connector not tunneling

Default log facility: discard

Argument	Description
Connector	Connector action
DNS entry	DNS entry ID
Application	Application
Dst	Address
Dst Port	Port



# Index

## Symbols

\$HOME, 9  
 %APPDATA%, 9  
 %USERPROFILE%, 9  
 .ssh2, 242–243  
 <INSTALLDIR>, 9

## A

account  
     local, 58  
 active mode FTP, 102  
 agent forwarding, 91, 189  
 AIX  
     installation, 21  
     uninstallation, 30  
 APPDATA, 9  
 application  
     direct connection, 161  
 Application Data, 36, 151  
 application tunneling, 91  
 ASCII file transfer mode, 237  
 association  
     file type, 231, 243  
 audit messages, 345  
 authentication, 47, 58  
     certificate, 44, 53–54, 67, 150, 155  
     GSSAPI, 70, 112, 118, 131  
     host-based, 69  
     Kerberos, 70  
     keyboard-interactive, 69, 118, 131  
     PAM, 69  
     password, 55, 69, 118, 131  
     public-key, 44, 48, 58, 118, 131, 150  
     RADIUS, 69  
     SecurID, 69  
 authentication methods, 47, 118, 130–131, 184  
 authority info access, 54  
 authorization file, 255  
 authorized\_keys directory, 254

authorized\_keys file, 255  
 automatic tunnels, 166

## B

backup files, 206  
 base-64, 336  
 basic configuration, 45, 115  
 binary file transfer mode, 237

## C

C-API, 11  
 CA certificate, 54, 155, 174  
 capture, 160–161  
 case-sensitivity, 231, 280, 306  
 certificate  
     enrolling, 68  
     revoked, 54  
 certificate authentication  
     server, 53–54, 155, 172  
     user, 67, 150  
 certificate enrollment, 69  
 certificate revocation list  
     prefetch, 159  
 certificate revocation list (CRL), 54–55, 174  
 certificate validation, 172  
 certificate viewer, 334  
 certificates, 150, 218  
 certification  
     FIPS 140-2, 117, 172  
 certification authority (CA), 53, 172  
 CertKey, 68  
 channel, 91  
 characters  
     valid, 280, 306  
 checkpoint-restart, 292  
 chmod, 235  
 ciphers, 183  
 client configuration file, 33  
 CMP client, 324  
 color settings, 144, 226, 228  
 command-line options, 46, 241  
 command-line tools, 249

- components, 39
- compression, 187
- configuration, 115
- configuration file, 168
  - server, 33
  - syntax, 207
- configuration file backup, 206
- configuration files, 45
- configuring, 115
- configuring menus, 244
- configuring terminal window, 221
- confirmation dialogs, 228
- Connection Broker, 44–45, 115, 168
  - debugging, 109
- connection capture, 160–161
- connection log, 218
- connection profile, 194
- connection profiles, 128
- connection settings, 115
- connections, 217
- Connections view, 217
- Control Panel, 32
- controlling file transfer, 79
- copying text, 223
- create shortcut, 129
- CRL
  - disabling, 55, 174
  - prefetch, 159
- CRL distribution point, 54
- CRL prefetch, 174
- cryptographic library, 117, 172
- customer support, 9
- customizing
  - pop-up menus, 243
  - shortcut menus, 243
  - toolbars, 248
- customizing settings, 243

## D

- default domain, 157, 173
- default installation directory, 27
- default profile, 242

## defining

- TCP tunnels, 160
- defining Connection Broker menu items, 117
- defining date format, 231
- defining pop-up menus, 246
- defining shortcut menus, 246
- defining terminal colors, 226
- defining time format, 231
- defining user interface settings, 221
- deleting remote folders, 78
- desktop, 28, 243
- Diffie-Hellman key exchange, 48, 53
- digital signature, 58
- direct connection
  - applications, 161
- directory
  - default installation, 27
  - root directory, 230
- disabling CRL, 55, 156, 174
- disk space requirement, 17
- dns, 203
- Document Type Definition (DTD), 207
- documentation, 7
- documentation conventions, 8
- DoD PKI, 157, 174
- domain user account, 67
- DOS shell, 241
- download status, 76
- downloading files, 75
- downloading software, 20
- dynamic tunnels, 103

## E

- editing configuration files, 46
- egrep syntax, 341
- end-point identity check, 157, 172
- enrolling certificates, 69
- enrolling user certificate, 68
- Entrust, 152
- Entrust keys, 182
- environment variables, 9, 269, 282, 308
- euro character, 147

event log, 126, 206  
exclusive-connection, 189  
exit values  
    scpg3, 283  
    sftpg3, 310  
    sshg3, 270  
expired CRL, 174  
external key viewer, 338

## F

Federal Information Processing Standard (FIPS), 117, 172  
file locations  
    installed files, 33  
file permissions, 234  
file size, 78  
file transfer, 73–74, 284, 311  
    controlling, 79  
    downloading, 75  
    mode, 236–237  
    uploading, 76  
file transfer settings, 148, 229, 233  
file transfer window default view, 230  
file type association, 231, 243  
file-access-control, 179  
filename characters, 280, 306  
filename support, 280, 306  
filter, 203  
filter engine, 169, 202  
filter rules, 162  
fingerprint, 48, 317–319  
FIPS 140-2 certification, 117, 172  
FIPS mode, 183–184  
firewall, 54  
folder  
    root directory, 230  
font  
    size, 225  
font size, 225  
fonts  
    installed, 225  
    terminal, 224

forwarding  
    agent, 91  
    agent forwarding, 123, 140  
    local, 91  
    remote, 104  
    X11, 91, 123, 140  
forwarding X11, 106  
FTP active mode, 102  
FTP passive mode, 102

## G

generating keys, 64, 150  
Generic Security Service API (GSSAPI), 70  
getting started, 39  
glob patterns, 303  
global settings, 221  
global.dat, 221, 242  
GSSAPI authentication, 70, 112, 118, 131  
    troubleshooting, 112  
GUI type, 143

## H

hardware requirement, 17  
hashed host key format, 49  
Hexl, 336  
hidden files, 230  
HOME, 9  
host key, 52  
    directory, 252–253  
    hashed format, 49  
    managing, 153  
    public, 48  
    resolving, 52  
host key check, 190  
host settings, 39  
host-based authentication, 69  
hostbased default domain, 187  
hostkeys directory, 252–253  
hostname, 40  
HP-UX  
    installation, 22  
    uninstallation, 31

HTTP proxy URL, 157, 173

HTTP repository, 54

## I

IBM AIX, 21

icons, 28, 243

identification file, 60, 68, 252

IdKey, 60

idle timeout, 188

incoming tunnels, 142

installation

removing, 30

silent, 27

upgrading, 19

installation directory, 27

INSTALLDIR, 9

installed files, 33

installed fonts, 225

installing

on Linux on IBM System z, 29

installing on AIX, 21

installing on HP-UX, 22

installing on Linux, 23

installing on Solaris, 23

installing on VMware ESX, 28

installing on Windows, 25

installing on z/Linux, 29

installing SSH Tectia Client, 21

## J

Java API, 11

## K

keepalive messages, 189

keepalive-interval, 189

Kerberos authentication, 70

key exchange, 48, 53

key file, 64, 133

key fingerprint, 48, 317–319

key pair, 58

key providers, 151

key security, 58

key stores, 175, 182

keyboard shortcut, 245

keyboard-interactive authentication, 69, 118, 131

keys, 150, 153, 218

Keys view, 218

known\_hosts file, 53, 180, 254

## L

LDAP servers, 158, 173

library

cryptographic, 117, 172

library certification

FIPS 140-2, 117, 172

license file, 17

licensing, 17

Lightweight Directory Access Protocol (LDAP), 54

Linux

installation, 23

uninstallation, 31

Linux on IBM System z

installation, 29

uninstallation, 33

local port forwarding, 91

local tunnel, 91

local tunnels, 141

local user account, 58

locale, 231

location

installed files, 33

log information, 218

logging, 126, 206, 218

Logs view, 218

## M

MACs, 184

maintenance release, 20

man pages, 249

man-in-the-middle attack, 48, 53

maximum file size, 78

menu options, 117, 162, 216–217

moving, 244

message, 228

- Microsoft Crypto API, 151
- Microsoft Office XP look, 223
- Microsoft Windows, 25
- modifying configuration files, 46
- moving menu options, 244
- moving toolbars, 248
- MSCAPI, 151
- MSI package, 25
- multiple windows, 242

## N

- nested tunnel, 130
- network, 203
- non-interactive installation, 27
- notation
  - path, 280, 306

## O

- OCSP responders, 156, 174
- Online Certificate Status Protocol (OCSP), 54
- online purchase, 17
- OpenSSH authorized\_keys file, 255
- OpenSSH keys, 67, 182
- OpenSSH known\_hosts file, 180, 254
- options
  - command-line, 241
- outgoing tunnels, 141

## P

- packaging, 17
- PAM authentication, 69
- passive mode FTP, 102
- passphrase, 59
- password
  - stored, 56
  - window out of focus, 112
- password authentication, 55, 69, 118, 131
- path notation, 280, 306
- pattern syntax, 341
- PEM encoding, 336
- permissions, 58
- PKCS #11 token, 68

- PKCS #12 certificates, 69
- PKCS #7 certificates, 69
- PKCS #7 package, 54
- PKCS#11 keys, 182
- PKCS#12, 182
- PKCS#7, 182
- Pluggable Authentication Module (PAM), 69
- pop-up menus, 76, 246
- port forwarding, 91, 139
  - local, 91
  - remote, 104
  - restricting, 91
- port number, 40
- positioning menu items, 244
- printing, 239
- private key
  - user, 59, 69
- profile
  - roaming, 58
- profile settings, 128
- profiles
  - default, 242
- program group, 28
- program icon, 28
- program shortcuts, 243
- Programs menu, 28
- proxy rules, 187
- proxy settings, 124, 138
- public key, 63
  - host, 48, 153
  - user, 59
- public-key authentication, 44, 58
  - server, 48, 153
  - user, 58, 118, 131, 150
- Public-Key Authentication Wizard, 63

## R

- RADIUS authentication, 69
- random\_seed file, 252
- Red Hat Linux, 23, 29
- regex syntax, 341
- registry keys, 36

- regular expression (regex), 280, 306, 341
- regular expressions, 341
- rekey interval, 184
- related documents, 7
- remote environment, 193
- remote folders
  - deleting, 78
- remote port forwarding, 104
- remote tunnel, 104
- remote tunnels, 142
- removing
  - from Linux on IBM System z, 33
  - old versions, 19
- removing from AIX, 30
- removing from HP-UX, 31
- removing from Linux, 31
- removing from Solaris, 31
- removing from VMware ESX, 32
- removing from Windows, 32
- removing SSH Tectia Client, 30
- return values
  - scp3, 283
  - sftp3, 310
  - ssh3, 270
- revoked certificate, 54
- RFC 4253, 322
- RFC 4716, 323
- roaming profile, 58
- RPM packages, 23, 29
- rule, 204

## S

- SAF authentication
  - server, 183
  - user, 183
- SCEP client, 331
- scp3, 74, 271
  - environment variables, 282
  - exit values, 283
- secure application connectivity, 91
- secure copy (SCP), 74, 271
- secure file transfer, 73

- Secure File Transfer Protocol (SFTP), 74, 284
- Secure Shell version 2, 261
- secured connections, 217
- SecurID authentication, 69
- security issues, 58
- server authentication, 152
- server authentication with certificates, 53–54, 155
- server authentication with public key, 48, 153
- server certificate, 53
- settings
  - file, 242
  - file transfer, 229, 233
  - host, 39
  - on Windows, 115
  - profile, 128
  - saving, 242
  - upload, 234
  - user interface, 221
- SFTP
  - checkpoint, 292
  - streaming, 292
- sftp3, 74, 284
  - commands, 288
  - environment variables, 308
  - exit values, 310
- shortcut menus, 76, 246
- silent installation, 27
- smart card, 68
- SOCKS server, 103
- SOCKS server URL, 157, 173
- Solaris
  - installation, 23
  - uninstallation, 31
- sorting order, 231
- SSH Tectia Client, 10
- SSH Tectia Client components, 39
- SSH Tectia Connections Configuration tool, 115
- SSH Tectia ConnectSecure, 11
- SSH Tectia icon, 28
- SSH Tectia MFT Events, 11
- SSH Tectia Server, 11
- SSH Tectia Server for IBM z/OS, 11
- SSH Tectia Server for Linux on IBM System z, 11

SSH Tectia Status, 217  
 ssh-broker-config.xml, 45, 168  
 ssh-broker-ctl, 255  
     commands, 256  
 ssh-broker-g3, 250  
 ssh-certview-g3, 334  
 ssh-client-g3, 241  
 ssh-cmpclient-g3, 324  
 ssh-ekview-g3, 338  
 ssh-keyfetch, 321  
 ssh-keygen-g3, 59, 316  
 ssh-scepclient-g3, 331  
 ssh-translation-table, 311  
 ssh-troubleshoot, 110, 259  
     commands, 260  
 SSH2, 261  
 SSH2 keys, 182  
 ssh\_known\_hosts file, 254  
 sshg3, 261  
     environment variables, 269  
     exit values, 270  
 start menu, 28  
 status  
     download, 76  
     upload, 76  
 Status dialog box, 217  
 streaming, 292  
 strict host key checking, 190  
 Sun Solaris, 23  
 support, 9  
 supported platforms, 13  
 SUSE Linux, 23  
 SUSE LINUX, 29  
 system configuration, 115  
 system log, 126, 206  
 system message, 228  
 system requirements, 13

## T

taskbar icon, 216–217  
 TCP connection  
     keepalive, 189

    timeout, 189  
 tcp-connect timeout, 189  
 technical support, 9  
 terminal  
     fixed window size, 224  
 terminal answerback, 147  
 terminal colors, 226  
 terminal fonts, 224  
 terminal settings, 146  
 terminal window  
     fixed size, 225  
     height, 226  
     width, 226  
 terminology, 10  
 test connection, 128  
 time stamp, 234  
 timeout  
     TCP connection, 189  
 toolbars, 244, 248  
 transfer mode, 236  
 transparent TCP tunneling, 93  
 transport distribution, 184  
 tray icon, 117, 216–217  
 tray menu, 117, 162, 216  
 troubleshooting, 109  
     password, 112  
 troubleshooting tool, 110  
 tunnel  
     local, 91  
     remote, 104  
 tunneling, 91, 139, 160  
     agent forwarding, 123, 140  
     applications, 103  
     restricting, 91  
     X11, 123, 140  
 tunneling e-mail, 93  
 tunneling X11, 106  
 tunnels  
     automatic, 166  
     incoming, 142  
     local, 141  
     outgoing, 141  
     remote, 142

## U

### uninstalling

- from Linux on IBM System z, 33
- uninstalling from AIX, 30
- uninstalling from HP-UX, 31
- uninstalling from Linux, 31
- uninstalling from Solaris, 31
- uninstalling from VMware ESX, 32
- uninstalling from Windows, 32
- uninstalling SSH Tectia Client, 30
- upgrading, 19
- upload status, 76
- uploading a public key, 65, 150
- uploading files, 76
- uploading public keys, 60
- uploading settings, 234
- user account
  - domain, 67
  - local, 58
- user authentication based on host, 69
- user authentication with certificates, 67, 150
- user authentication with GSSAPI, 70
- user authentication with keyboard-interactive, 69
- user authentication with password, 55
- user authentication with public key, 58, 150
- user certificate
  - enrolling, 68
- user configuration directory, 178
- user identity, 195
- user key, 60, 63
- user name, 40
- user-config-directory, 178
- USERPROFILE, 9
- using secure copy, 74
- using secure file transfer, 74

## V

- valid characters, 280, 306
- viewing key and certificate information, 218
- viewing log information, 218
- viewing status, 217
- viewing tunnel information, 217

- VMware ESX, 28
  - installation, 28
  - uninstallation, 32

## W

- wildcard, 237, 280, 303, 306
- window
  - multiple windows, 242
  - positions, 242
  - settings, 143
  - size, 225
- Windows
  - desktop, 28, 243
  - Event Log, 126
  - installation, 25
  - password, 55
  - registry keys, 36
  - Start menu, 28
  - taskbar, 216
  - uninstallation, 32
- Windows Explorer, 78, 231

## X

- X.509 certificate, 54, 68
- X.509 certificates, 69, 182
- X11 forwarding, 91, 106, 140, 189
- XML attribute
  - allow-relay, 198, 201
  - data, 196
  - default-domain, 173
  - disable-crls, 174
  - end-point-identity-check, 172
  - file, 196
  - gateway-profile, 195
  - hash, 196
  - http-proxy-url, 173
  - id, 196
  - identity-file, 196
  - socks-server-url, 173
  - use-expired-crls, 174
- XML element
  - accept-unknown-host-keys, 178



---

auth-gssapi, 186  
auth-hostbased, 185  
auth-keyboard-interactive, 186  
auth-password, 185  
auth-publickey, 185  
auth-server-certificate, 190  
auth-server-publickey, 190  
authentication-method, 185, 192  
authentication-methods, 184, 195  
authentication-success-message, 193  
ca-certificate, 174  
cert-validation, 172  
cipher, 183  
ciphers, 183, 195  
compression, 187, 196  
crl-prefetch, 174  
crypto-lib, 172  
default-settings, 183  
dns, 203  
dod-pki, 174  
environment, 193  
exclusive-connection, 189, 197  
file-access-control, 179  
filter, 203  
filter-engine, 202  
forward, 189  
forwards, 189, 197  
general, 172  
host-key-always-ask, 177  
hostbased-default-domain, 187  
hostkey, 195  
identification, 177  
identity, 196  
idle-timeout, 188, 196  
keepalive-interval, 189, 197  
key-selection, 186  
key-store, 175–176, 182  
key-stores, 175  
known-hosts, 180  
ldap-server, 173  
local-hostname, 185  
local-tunnel, 197  
log-events, 206  
logging, 206  
mac, 184  
macs, 184, 195  
network, 203  
ocsp-responder, 174  
password, 199  
profile, 194  
profiles, 194  
proxy, 187, 196  
public-key, 186  
rekey, 184, 195  
remote-environment, 193, 199  
remote-tunnel, 198  
rule, 204  
server-authentication-methods, 199  
server-banners, 189, 197  
sftp3-mode, 193  
static-tunnels, 200  
strict-host-key-checking, 177  
tcp-connect-timeout, 189, 197  
transport-distribution, 184, 195  
tunnel, 200  
tunnels, 197  
user-config-directory, 178  
user-identities, 195  
user-keys, 176

