SSH.COM

Tectia® Server 6.6

Administrator Manual

04 December 2024

Tectia® Server 6.6: Administrator Manual

04 December 2024

SSH Communications Security Corporation
Karvaamokuja 2D, FI-00380 Helsinki, Finland

Table of Contents

1. About This Document 9
1.1. Documentation Conventions 10
1.1.1. Operating System Names 10
1.1.2. Directory Paths 11
1.2. Customer Support 11
1.3. Component Terminology 12
2. Installing Tectia Server 15
2.1. Preparing for Installation 15
2.1.1. System Requirements 15
2.1.2. Hardware and Disk Space Requirements 16
2.1.3. Licensing 17
2.1.4. Installation Packages 17
2.1.5. Upgrading Previously Installed Tectia Server Software 18
2.1.6. Downloading Tectia Releases 21
2.2. Installing the Tectia Server Software 22
2.2.1. Installing on AIX 22
2.2.2. Installing on HP-UX 24
2.2.3. Installing on Linux (RPM) 25
2.2.4. Installing on Linux (DEB) 27
2.2.5. Installing on Solaris 28
2.2.6. Installing on Windows 30
2.3. Removing the Tectia Server Software 33
2.3.1. Removing from AIX 33
2.3.2. Removing from HP-UX 33
2.3.3. Removing from Linux 34
2.3.4. Removing from Solaris 34
2.3.5. Removing from Windows 35
2.4. Files Related to Tectia Server 35
2.4.1. File Locations and Permissions on Unix 36
2.4.2. File Locations on Windows 37
2.4.3. Registry Keys on Windows 39
3. Getting Started 41
3.1. Starting and Stopping the Server 41

3.1.1. Starting and Stopping on AIX 41
3.1.2. Starting and Stopping on Other Unix Platforms 42
3.1.3. Starting and Stopping on Windows 42
4. Configuring Tectia Server 45
4.1. Tectia Server Configuration Tool 45
4.1.1. Tectia Server 46
4.1.2. General 49
4.1.3. Proxy Rules 54
4.1.4. Domain Policy 55
4.1.5. Password Cache 57
4.1.6. Identity 60
4.1.7. Network 62
4.1.8. Logging 64
4.1.9. Certificate Validation 65
4.1.10. Defining Access Rules Using Selectors (Advanced Mode) 69
4.1.11. Connections and Encryption 76
4.1.12. Authentication 80
4.1.13. Services 88
4.2. Configuration File for Tectia Server 102
4.2.1. Dividing the Configuration into Several Files 103
4.2.2. Using Selectors in Configuration File 104
4.2.3. ssh-server-config.xml 108
5. Authentication 167
5.1. Supported User Authentication Methods 167
5.1.1. Compatibility with OpenSSH Keys and Certificates 168
5.2. Server Authentication with Public Keys 168
5.2.1. Generating the Host Key 169
5.2.2. Notifying the Users of Host Key Changes 170
5.2.3. Key rotation 171
5.3. Server Authentication with Certificates 171
5.3.1. Certificate Enrollment Using `ssh-cmpclient-g3` 172
5.4. Server Authentication Using External Host Keys 173
5.5. User Authentication with Passwords 173
5.5.1. Expired Passwords 174
5.5.2. Empty/Blank Passwords 174
5.5.3. User Logon Rights on Windows 174
5.6. User Authentication with Public Keys 175
5.6.1. Using the Authorization File 175
5.6.2. Using Keys Generated with OpenSSH 176
5.6.3. Special Considerations on Windows 176
5.6.4. Authorized Keys on a Windows Network Drive 177
5.7. User Authentication with Certificates 178
5.7.1. Configuring Certificates 178
5.7.2. Configuring User Authentication with Certificates on Windows 181
5.8. Host-Based User Authentication 188

5.8.1. Using Conventional Public Keys 188
5.8.2. Using Certificates 190
5.9. User Authentication with Keyboard-Interactive 193
5.9.1. Password Submethod 193
5.9.2. Pluggable Authentication Module (PAM) Submethod 194
5.9.3. RSA SecurID Submethod 197
5.9.4. RADIUS Submethod 199
5.9.5. LAM Submethod on AIX 200
5.10. User Authentication with GSSAPI 201
5.11. Supplementing Authentication with an External Application 202
5.11.1. Example with Certificate Authentication 202
5.11.2. Example with Password Authentication 203
5.12. Configuring User Authentication Chains 203
5.12.1. Basic Example 203
5.12.2. Example with Selectors 204
5.12.3. Authentication Chain Example 204
5.12.4. Example of Using the Deny Action 205
5.13. Forwarding User Authentication 206
5.13.1. Forwarding User Authentication to a Kerberos Realm 206
5.14. Reporting User Login Failures 207
5.15. User Name Handling on Windows 208
5.16. Requirements for Trusted Domain Authentication on Windows 209
5.17. Accessing Resources on Windows Network from Logon Sessions Created by Tectia Server .. 210
5.17.1. Network Resource Access from Terminal Session 213
5.17.2. Network Resource Access from SFTP Subsystem 215
5.17.3. Accessing Network Shares Using Another User's Account 215
5.17.4. Accessing Shares on a Computer That Is Not a Member of a Domain 216
5.17.5. Access to DFS Shares 216
5.18. Accessing Files Stored on EFS on Windows from Logon Sessions Created by Tectia Server . 216
6. System Administration 219
6.1. Tectia Client Privileged User 219
6.1.1. Disabling Root Login (Unix) 219
6.1.2. Restricting Connections 220
6.1.3. Chrooting (Unix) 220
6.1.4. Forced Commands 222
6.2. Auditing 223
6.2.1. Notification 224
6.2.2. Customizing Logging 224
6.2.3. Auditing with Solaris BSM 226
7. File Transfer 227
7.1. Tectia Client File Transfer User 228
7.1.1. Encryption and Authentication Methods 228
7.1.2. Restricting Services 228
7.1.3. Settings on the Client Side 233
7.2. Automated File Transfer Script 234

8. Tunneling 235
8.1. Transparent TCP Tunneling from Server Perspective 235
8.1.1. Using a Shared Account 236
8.1.2. Restricting Services 236
8.2. Local Tunnels 238
8.2.1. Local Tunneling Rule Examples 239
8.3. Remote Tunnels 243
8.3.1. Remote Tunneling Rule Examples 244
8.4. X11 Forwarding (Unix) 245
8.5. Agent Forwarding (Unix) 247
9. Troubleshooting Tectia Server 249
9.1. Starting Tectia Server in Debug Mode 249
9.1.1. Enabling Debug Mode for Running Tectia Server on Unix 249
9.1.2. Starting Tectia Server in Debug Mode on Unix 250
9.1.3. Enabling Debug Mode for Running Tectia Server on Windows 251
9.1.4. Starting Tectia Server in Debug Mode on Windows 252
9.1.5. Debugging Secure File Transfer 255
9.2. Collecting System Information for Troubleshooting 256
9.3. Solving Problem Situations 257
9.3.1. CPU Overload on Tectia Server on HP-UX 257
9.3.2. Invalid Host Key Permissions on Windows 257
9.3.3. Invalid Configuration File Permissions on Windows 258
9.3.4. Authentication Fails for Domain Account on Tectia Server on Windows 258
9.3.5. Last Login Time is Incorrect on Windows 259
9.3.6. Virtual Folders Defined on Windows Network Shared Folders Are Not Available on Tectia Server on Windows 259
A. Tectia Server Configuration File Quick Reference 261
B. Server Configuration File Syntax 275
C. Command-Line Tools and Man Pages 293
ssh-server-g3 294
ssh-server-ctl 304
ssh-troubleshoot 313
ssh-keygen-g3 315
ssh-keyfetch 321
ssh-cmpclient-g3 325
ssh-scepclient-g3 332
ssh-certview-g3 336
ssh-ekview-g3 340
D. Audit Messages 341
E. Tectia Mapper Protocol 399
E.1. Parameters 399
E.2. Communication Between Tectia Server and the External Application 400
E.3. Examples 401
E.3.1. Positive Response 401
E.3.2. Negative Response 402

E.3.3. Checking the Number of Connections 402
E.4. Example Application 403
F. Removing OpenSSL from Tectia Server 405
F.1. Background Information 405
F.1.1. Should I Remove the OpenSSL Library? 405
F.1.2. What Happens If I Remove the OpenSSL Library? 405
F.2. Removing the OpenSSL Cryptographic Library 405
F.2.1. Linux and Solaris 405
F.2.2. Windows 406
G. Default and Supported SSH Algorithms 409
G.1. Ciphers 409
G.2. Key-Exchange Algorithms 410
G.3. Message-Authentication Codes 411
G.4. Host-Key and Public Key Signature Algorithms 412
H. Open Source Software License Acknowledgements 415
I. Changing the Host Key of Tectia Server 423
I.1. Host key Algorithm in Manual Host Key Rotation 424
I.2. Manual Rotation Example using RSA Host Keys 425
I.3. Fingerprints 428
I.4. Replacing Host Public Key on Client-Side 428
I.4.1. z/OS Example 429
I.4.2. Windows Tectia Client Example 429
Index 431

Chapter 1 About This Document

This document contains instructions on the basic administrative tasks of Tectia Server. It is intended for system administrators responsible for the configuration of the Tectia Server software.

There are two separate Tectia Server product versions:

- Tectia Server (for Unix and Windows)
- Tectia Server for IBM z/OS

Tectia Server for Unix and Windows is handled in this manual.

Tectia Server for IBM z/OS is released separately and described in its own Administrator Manual.

This document contains the following information:

- Installing Tectia Server
- Getting started
- Configuring Tectia Server
- Authentication settings
- System administration
- File transfer
- Tunneling
- Troubleshooting
- Appendices, including command-line tool and audit message references

To fully use the information presented in this document, you should be familiar also with other system administration tasks. To edit the configuration files manually without Tectia Server Configuration GUI, you should have basic knowledge of XML.

Tectia Client/Server Product Description contains important background information on the Tectia client/server solution, and we recommend that you read it before installing and starting Tectia Server.

1.1 Documentation Conventions

The following typographical conventions are used in Tectia documentation:

Table 1.1. Documentation conventions

Convention	Usage	Example
Bold	Tools, menus, GUI elements and commands, command-line tools, strong emphasis	Click **Apply** or **OK**.
→	Series of menu selections	Select **File** → **Save**
`Monospace`	Command-line and configuration options, file names and directories, etc.	Refer to `readme.txt`
Italics	Reference to other documents or products, URLs, emphasis	See *Tectia Client User Manual*
`Monospace Italics`	Replaceable text or values	rename *`oldfile newfile`*
#	In front of a command, # indicates that the command is run as a privileged user (root).	`# rpm --install package.rpm`
$	In front of a command, $ indicates that the command is run as a non-privileged user.	`$ sshg3 user@host`
\	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.	`$ ssh-keygen-g3 -t rsa \ -F -c mykey`

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
  - Solaris
  - IBM z/OS, when applicable; as 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:\Users\<username>\AppData\Roaming`".

`%USERPROFILE%`

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

"`C:\Users\<username>`".

`<INSTALLDIR>`

Indicates the default installation directory on Windows:

"`C:\Program Files (x86)\SSH Communications Security\SSH Tectia`" on 64-bit Windows versions

1.2 Customer Support

All Tectia product documentation is available at https://www.ssh.com/manuals/.

FAQ with how-to instructions for all Tectia products are available at https://documents.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 https://support.ssh.com/.

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 Tectia Client, Tectia ConnectSecure, and in the Tectia Server for IBM z/OS client tools. Connection Broker handles all cryptographic operations and authentication-related tasks.

FTP-SFTP conversion
: Tectia ConnectSecure can automatically capture FTP connections on the client and convert them to SFTP and direct them to an SFTP server running Tectia Server, Tectia Server for IBM z/OS, or another vendor's Secure Shell server software.

host key pair
: A public-key pair used to identify a Secure Shell server. The private hostkey 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, client computer or server computer, depending on the viewpoint.

Secure Shell client
: A client-side application that uses the Secure Shell version 2 protocol, for example **sshg3**, **sftpg3**, or **scpg3** of 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.

Tectia Client
: A software component installed on a workstation. Tectia Client provides secure interactive file transfer and terminal client functionality for remote users and system administrators to access and manage servers running Tectia Server or other applications using the Secure Shell protocol. It also supports (non-transparent) static tunneling.

Tectia client/server solution
: The Tectia client/server solution consists of Tectia Client, Tectia ConnectSecure, Tectia Server, and Tectia Server for IBM z/OS (including the Tectia Server for IBM z/OS client tools).

Tectia Connections Configuration GUI
: Tectia Client and ConnectSecure have a graphical user interface for configuring the connection settings to remote servers. The GUI is supported on Windows and Linux.

Tectia ConnectSecure
: A software component installed on a server host, but it acts as a Secure Shell client. Tectia ConnectSecure is designed for FTP replacement and it provides FTP-SFTP conversion, transparent FTP tunneling, transparent TCP tunneling, and enhanced file transfer services. Tectia ConnectSecure is capable of connecting to any standard Secure Shell server.

Tectia Secure File Transfer GUI
: Tectia Client and ConnectSecure on Windows include a separate graphical user interface (GUI) for handling and performing file transfers interactively.

Tectia Server
: Tectia Server is a server-side component where Secure Shell clients connect to. There are two versions of the Tectia Server product available: *Tectia Server* for Linux, Unix and Windows platforms, and *Tectia Server for IBM z/OS*.

Tectia Server for IBM z/OS
: 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.

Tectia Server Configuration tool
: Tectia Server has a graphical user interface that can be used to configure the server instead of editing the configuration file. The GUI is supported on Windows.

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 Tectia Server

This chapter contains instructions on installing (and removing) Tectia Server on the supported Unix, Linux, and Windows platforms.

2.1 Preparing for Installation

This section lists the supported platforms and gives the necessary prerequisites for the Tectia Server installation.

Note

If planning to use FIPS (Federal Information Processing Standard) cryptolibrary, FIPS MODE should be enabled prior to installation to ensure Tectia Server generates default server host key upon installation in FIPS mode or the hostkey needs to be regenerated manually. Please see **crypto-lib** for more information.

2.1.1 System Requirements

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

Table 2.1. Supported operating systems for Tectia Client and Server

Operating System	Client	Server
HP-UX (PA-RISC) [a]	11i v3	11i v3
HP-UX (IA-64) [a]	11i v3	11i v3
IBM AIX (POWER)	7.1, 7.2, 7.3	7.1, 7.2, 7.3
Oracle Solaris (SPARC)	10, 11	10, 11
Oracle Solaris (x86-64)	10, 11	10, 11
Red Hat Enterprise Linux (x86-64)	7, 8, 9	7, 8, 9
Rocky Linux (x86-64)	8, 9	8, 9

Operating System	Client	Server
Ubuntu (x86-64)	18.04, 20.04, 22.04	18.04, 20.04, 22.04
Debian GNU/Linux (x86-64)	11, 12	11, 12
SUSE LINUX Enterprise Desktop (x86-64)	15	15
SUSE LINUX Enterprise Server (x86-64)	12, 15	12, 15
Microsoft Windows (x86-64)	10, 11, Server 2016, Server 2019, Server 2022	10, 11, Server 2016, Server 2019, Server 2022

[a] for this platform PQC is not supported.

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. Tectia solutions have been tested with the following patches and maintenance levels:

HP-UX patches

The general principle is to install the latest **HP-required patch bundle** for the OS version. Proper functioning of the 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 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

Tectia Server 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 TCP/IP network connection can be used.

Tectia Server requires disk space as follows:

- 1 GB RAM for hundreds of simultaneous tunnels
- 100 MB free disk space

2.1.3 Licensing

Tectia Server requires a license to function. The license file is named `sts66.dat`.

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

- In the commercial installation packages, the license file(s) are included in the compressed (`.zip`/`.tar`) files together with the release notes (`.txt`) files and the PDF-format documentation.
- The Tectia evaluation packages do not contain license files; the evaluation versions can be used for 45 days without a license file. On Unix and Windows machines, a banner message will remind users of how many days are left until the license expires.
- When upgrading the evaluation version or standard commercial version to Tectia Quantum Safe Edition only license file(s) need to be copied to the license directory and Tectia Server software restarted.

2.1.4 Installation Packages

The installation packages of Tectia Server are compressed into installation bundles. There are three bundles for each supported operating system, the Tectia Quantum Safe Edition commercial version (`-comm-pqc`), 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 Tectia Server bundle:

- For AIX platforms:

```
tectia-server-<version>-aix-6-7-powerpc-comm-pqc.tar
tectia-server-<version>-aix-6-7-powerpc-comm.tar
tectia-server-<version>-aix-6-7-powerpc-upgrd-eval.tar
```

- For HP-UX PA-RISC platforms:

```
tectia-server-<version>-hpux-11iv2-3-hppa-comm-pqc.tar
tectia-server-<version>-hpux-11iv2-3-hppa-comm.tar
tectia-server-<version>-hpux-11iv2-3-hppa-upgrd-eval.tar
```

- For HP-UX Itanium platforms:

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

- For Linux 64-bit platforms (Red Hat Enterprise Linux, Rocky Linux and SUSE Linux):

```
tectia-server-<version>-linux-x86_64-comm-pqc.tar
```

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

- For Linux 64-bit platforms (Ubuntu and Debian GNU/Linux):

```
tectia-server-<version>-linux-ubuntu-x86_64-comm-pqc.tar
tectia-server-<version>-linux-ubuntu-x86_64-comm.tar
tectia-server-<version>-linux-ubuntu-x86_64-upgrd-eval.tar
```

- For Solaris SPARC platforms (note the separate packages for Solaris 10 and 11):

```
tectia-server-<version>-solaris-10-sparc-comm.tar
tectia-server-<version>-solaris-10-sparc-upgrd-eval.tar
tectia-server-<version>-solaris-11-sparc-comm.tar
tectia-server-<version>-solaris-11-sparc-upgrd-eval.tar
```

- For Solaris x86-64 platforms (note the separate packages for Solaris 10 and 11):

```
tectia-server-<version>-solaris-10-x86_64-comm.tar
tectia-server-<version>-solaris-10-x86_64-upgrd-eval.tar
tectia-server-<version>-solaris-11-x86_64-comm.tar
tectia-server-<version>-solaris-11-x86_64-upgrd-eval.tar
```

- For Windows platforms:

```
tectia-server-<version>-windows-comm-pqc.zip
tectia-server-<version>-windows-comm.zip
tectia-server-<version>-windows-upgrd-eval.zip
```

`<version>` indicates the product release version and the current build number (for example `6.6.5.123`).

Inside the installation bundles are the actual installation packages for Tectia Server. On Unix and Linux platforms, the Tectia Server has the following installation packages:

- the `ssh-tectia-common` package contains the common components of Tectia Client and Server.
- the `ssh-tectia-server` package contains the specific components of Tectia Server.
- the `ssh-tectia-client` package contains the specific components of Tectia Client.
- on Linux only, the `ssh-tectia-guisupport` RPM package contains the specific components of Tectia Connections Configuration GUI.

On Windows, Tectia Server comes in a single MSI installation package.

2.1.5 Upgrading Previously Installed Tectia Server Software

Note

Before starting the upgrade, make backups of all configuration files where you have made modifications.

When upgrading a maintenance release of Tectia Server on Windows, usually no rebooting of the computer is needed. Check the release notes to see if the current Server release can be upgraded without reboot. On Unix, upgrading does not require a reboot.

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

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

Before installing 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.

When upgrading on SUSE, also install the prerequisite packages:

```
# zypper install insserv-compat
```

The following table shows you which Tectia versions you need to uninstall before you can upgrade to Tectia Server 6.6. When upgrading versions marked *upgrade on top*, the earlier version is automatically removed during the upgrade procedure.

Table 2.2. Upgrade lines

Tectia version	AIX	HP-UX	Linux	Solaris	Windows
4.x	remove	remove	remove	remove	remove
5.x-6.0	upgrade on top	upgrade on top	upgrade on top	remove	remove
6.1-6.6	upgrade on top	upgrade on top	upgrade on top	remove	upgrade on top

The configuration file format and file locations have been changed in Tectia Server 5.0 and the Unix DTD directories in version 6.2. Because of this, the configuration files behave differently when upgrading from 4.x and from 5.x-6.1 compared to when upgrading from 6.2 and later versions.

- The 6.2-6.x configuration files are used by 6.6 as such and automatically taken into use.

Note

Any explicitly configured settings, for example Ciphers, MACs and KEXs will be retained when upgrading. These might include insecure algorithms such as SHA-1 in KEX, or in host key or public-key signature algorithms. Also, for example the Post Quantum Cryptography (PQC) Hybrid Key Exchange algorithms, that require the Tectia Quantum Safe Edition license, need to be prepended to any explicit KEX configuration(s) when upgrading from Tectia version 6.5 and below. Alternatively, the explicit configuration settings, for example all KEX algorithms, can be removed from the configuration to use the 6.6 defaults or the PQC hybrid KEX can be enforced.

- The 5.x-6.1 configuration files are used by 6.6 as such and on Windows platforms automatically taken into use.

Note

Any explicitly configured settings, for example Ciphers and MACs will be retained when upgrading. These might include insecure algorithms. In Tectia 6.1 and earlier on Unix the default auxiliary data directory `auxdata` was located in `/etc/ssh2/ssh-tectia/`. If your Tectia Server configuration file (`ssh-server-config.xml`) or Tectia Client configuration file (`ssh-broker-config.xml`) was created for Tectia version 6.1 or earlier, please update its DOCTYPE declaration to contain the current path to the server configuration file DTD directory: `/opt/tectia/share/auxdata/ssh-server-ng/` or the Connection Broker configuration file DTD directory: `/opt/tectia/share/auxdata/ssh-broker-ng/`.

- The 4.x configuration files are *not* migrated to 6.6, but the default 6.6 configuration is used. However, the connection profiles are migrated from 4.x to 6.6 on Windows platforms.

When necessary, you can modify the configuration files by using the Tectia Connections Configuration GUI or by editing the XML configuration files manually with an ASCII text editor or an XML editor. Please see example files `ssh-server-config-example.xml` for Tectia Server and `ssh-broker-config-example.xml` for Tectia Client.

Configuration File Access Permissions on Windows

When upgrading a previously installed version of Tectia Server on Windows, the access permissions for existing configuration files will be checked during the upgrade installation.

The access permissions for the `ssh-server-config.xml` configuration file should be as follows:

- The owner of the file is a member of the Administrators group.
- Only Administrators and SYSTEM may have full control of the file.
- Users are not allowed to modify the file.
- Other accounts do not have access to the file.

If the access permissions are not safe, you will see the **Configuration File Permissions** dialog box during the upgrade installation. Do one of the following:

- **Reset** the permissions for the configuration file to the default safe state and continue with the installation. (*Recommended*)
- **Ignore** the incorrect permissions and continue with the installation without fixing the permissions. Note that if you decide to do this, the server might not be able to start. You can fix the permissions manually later.
- **Cancel** the installation.

Note

Your previous installation of Tectia Server has already been removed, so if you cancel the installation, your machine will be left with no version of Tectia Server installed.

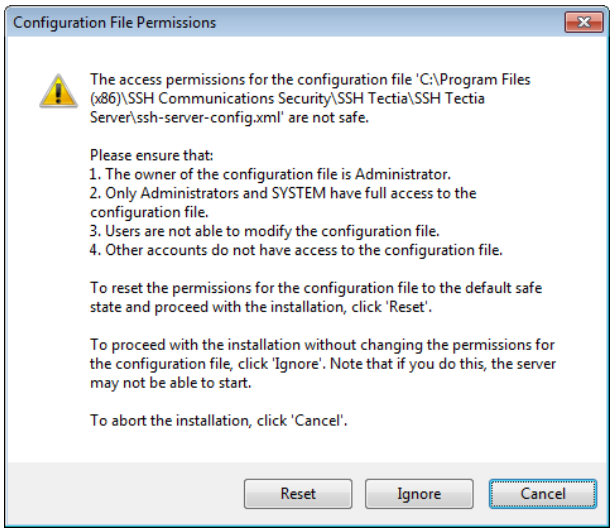


Figure 2.1. Unsafe configuration file permissions on Windows

Silent Upgrade on Windows

When doing a silent upgrade on Windows (see also the section called "Silent Installation") using the `/q` command-line option for `msiexec.exe`, the access permissions of an existing Tectia Server configuration file are checked. (The correct configuration file access permissions are described in the section called "Configuration File Access Permissions on Windows".) If the access permissions are incorrect, the server will, by default, be uninstalled.

To override the default behavior, specify the desired value (`1` or `2`) for the `SSHMSI_SSH_FILE_PERMISSIONS` property of the MSI installation package. Possible values are:

- `Cancel` or `0` (*default*) – abort the installation.
- `Reset` or `1` (*recommended*) – reset the configuration file access permissions to the default state.
- `Ignore` or `2` – continue the installation without modifying configuration file access permissions. Note that in this case the server and configuration utility may not be able to start until you fix the access permissions manually.

The following command can be used to upgrade Tectia Server silently in the default installation directory, resetting the configuration file access permissions to the default state:

```
msiexec /q /i ssh-tectia-server-<v>-windows-<p>.msi SSHMSI_SSH_FILE_PERMISSIONS=1
```

In the command, `<v>` is the current version of Tectia Server (for example, `6.6.5.123`), and `<p>` is the platform architecture (`x86_64` for 64-bit Windows versions).

2.1.6 Downloading Tectia Releases

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

To download Tectia software from the SSH Customer Download Center:

1. Log in to the Customer Download Center at: https://my.ssh.com
2. Select **Tectia Server** from the SSH Downloads, and choose the relevant version. Tectia products are published in major, minor, and maintenance releases:
   - Major releases are indicated with full numbers, for example 6. 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 6.6. Minor releases publish new features and fixes to the previous versions.
   - Maintenance releases are third digit versions, for example 6.6.5. 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 link with the correct product version and platform, 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 Tectia Server below.

2.2 Installing the Tectia Server Software

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

2.2.1 Installing on AIX

The downloaded installation package contains the compressed installation files.

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

If you are upgrading Tectia Server version 6.2.1 or earlier to 6.6, you must do the following steps before installing the new version:

1. Rename the subsystem group from `tcpip` to `ssh-tectia-server`:

```
# /usr/bin/rmssys -s ssh-tectia-server
```

2. Redefine `ssh-tectia-server` with the new group option:

```
# mkssys -s ssh-tectia-server -p "/opt/tectia/sbin/ssh-server-g3" -q -u 0 -S \
 -n 15 -f 9 -R -G ssh-tectia-server -i /dev/null -o /dev/null -e \
 /dev/null
```

3. Restart the ssh-tectia-server:

```
# stopsrc -s ssh-tectia-server
```

```
# startsrc -s ssh-tectia-server
```

Now you can continue with the installation steps.

Note that upgrading from Tectia Server version 6.2.x or 6.3.x will not restart the server automatically after installing the upgrade packages. Upgrading from Tectia Server versions 6.1.x (or earlier), and versions 6.4.2 (or later) will work normally and restart the server after upgrade.

To install Tectia Server on AIX, follow the instructions below:

1. Unpack the downloaded `tar` package.

2. Make sure no other software is using port 22 (Tectia Server default listen port). Stop any competing server software or change their listen port.

3. Unpack the installation packages:

   ```
   $ uncompress ssh-tectia-common-<version>-aix-6-7-powerpc.bff.Z
   $ uncompress ssh-tectia-server-<version>-aix-6-7-powerpc.bff.Z
   ```

   In the commands, *<version>* is the current package version of Tectia Server (for example, `6.6.5.123`).

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

   ```
   # installp -d ssh-tectia-common-<version>-aix-6-7-powerpc.bff SSHTectia.Common
   # installp -d ssh-tectia-server-<version>-aix-6-7-powerpc.bff SSHTectia.Server
   ```

   The server host key is generated during the initial installation. The key generation may take several minutes on slow machines.

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

   If this is the initial installation of Tectia Server, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

6. The installation should (re)start the server automatically.

Note

If you upgraded from Tectia Server 6.2.x or 6.3.x, the server will not restart automatically.

Note

If the server does not start (for example because of a missing license or because some other secure shell software is running on port 22), correct the problem and you can start the server process by using the System Resource Controller (SRC).

To start Tectia Server manually, enter command:

```
# startsrc -s ssh-tectia-server
```

Installing 32-bit LAM package for AIX

There is a 32-bit binary `ssh-aix-lam-proxy32` shipped with the Tectia Server installation package for AIX. In some cases there is a need to use a 32-bit Lightweight Authentication Module (LAM) in a 64-bit operating system, for example, when using Safeword authentication via LAM.

There are two binaries in `/opt/tectia/libexec`:

- ssh-aix-lam-proxy (64-bit binary)
- ssh-aix-lam-proxy32 (32-bit binary)

By default, the 64-bit binary is used. If the 32-bit binary is to be used, follow these steps:

1. Backup the `ssh-aix-lam-proxy` to a safe place.
2. Copy the `ssh-aix-lam-proxy32` to `ssh-aix-lam-proxy`.

This will automatically start using the 32-bit LAM on the 64-bit AIX host.

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 installation package contains the compressed installation files.

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

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

1. Unpack the downloaded `tar` package.
2. Make sure no other software is using port 22 (Tectia Server default listen port). Stop any competing server software or change their listen port.
3. Select the installation package according to your HP-UX version.

   When installing on HP-UX 11i v3 (11.31) running on the PA-RISC architecture, select the following packages:

   ```
   ssh-tectia-common-<version>-hpux-11iv2-3-hppa.depot.Z
   ssh-tectia-server-<version>-hpux-11iv2-3-hppa.depot.Z
   ```

   When installing on HP-UX 11i v3 running on the Itanium architecture, select the following packages:

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

In the commands, `<version>` indicates the product release version and the current build number (for example, `6.6.5.123`).

4. Unpack the installation packages with `uncompress`. In order to be installable, the created packages must have the correct long file names. In the following command examples, we use the Itanium packages:

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

5. 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-server-<version>-hpux-11i-ia64.depot SSHG3server
```

In the commands, `<path>` is the full path to the installation package.

Note

HP-UX requires the full path even when the command is run in the same directory.

The server host key is generated during the initial installation. The key generation may take several minutes on slow machines.

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

If this is the initial installation of Tectia Server, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

7. The installation should (re)start the server automatically.

Note

If the server does not start (for example because of a missing license or because some other secure shell software is running on port 22), you can start it after correcting the problem by issuing the command:

```
# /sbin/init.d/ssh-server-g3 start
```

2.2.3 Installing on Linux (RPM)

Tectia Server for Linux platforms is supplied in RPM (Red Hat Package Manager) binary packages for Red Hat Enterprise Linux, Rocky Linux and SUSE Linux running on the 64-bit architecture.

The downloaded installation package contains the RPM installation files.

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

To install Tectia Server on Linux, follow the instructions below:

1. If installing on SELinux-enabled systems, ensure that the `semanage` command is available. In older Linux versions `semanage` is typically installed via `policycoreutils-python-utils` or `policycoreutils-python`.

   Note

   On SELinux system, if an alternate port is used, for example "222" instead of the default secure shell port, use the following `semanage` command to allow it:

   ```
   # semanage port --add --type ssh_port_t --proto tcp 222
   ```

   If installing on SUSE, install prerequisite package:

   ```
   # zypper install insserv-compat
   ```

2. Unpack the downloaded `tar` package.

3. Make sure no other software is using port 22 (Tectia Server default listen port). Stop any competing server software or change their listen port.

4. Select the installation packages (in this example, we install Tectia Server only).

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

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

   In the commands, *<version>* indicates the product release version and the current build number (for example, `6.6.5.123`).

5. Install the packages with root privileges:

   ```
   # rpm -ivh ssh-tectia-common-<version>-linux-x86-64.rpm
   # rpm -ivh ssh-tectia-server-<version>-linux-x86-64.rpm
   ```

   The server host key is generated during the initial installation. The key generation may take several minutes on slow machines.

   Or upgrade the packages if you already have an older Tectia Server version installed:

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

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

   If this is the initial installation of Tectia Server, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

7. The installation should (re)start the server automatically.

Note

If the server does not start (for example because of a missing license or because some other secure shell software is running on port 22), you can start it manually after correcting the problem.

- Using Tectia Server control utility:

```
# ssh-server-ctl start
```

- Or on Linux with systemd:

```
# systemctl start ssh-server-g3
```

- Or on Linux without systemd:

```
# /etc/init.d/ssh-server-g3 start
```

2.2.4 Installing on Linux (DEB)

Tectia Server for Debian GNU/Linux platforms is supplied in Debian (DEB) binary packages for Ubuntu and Debian running on the 64-bit x86-64 architecture.

The Tectia Server installation bundle contains the DEB files and the license files for both the Tectia Server and Tectia Client that can be optionally installed on the same host.

To install Tectia Server on Debian, follow the instructions below:

1. Make sure no other Secure Shell software is using port 22 (Tectia Server default listen port). Also make sure the firewall is open for port 22.

2. Download the installation bundle according to your license type:

   - Commercial Tectia Quantum Safe Edition License:

     `tectia-server-<version>-linux-ubuntu-x86_64-comm-pqc.tar`

   - Commercial License:

     `tectia-server-<version>-linux-ubuntu-x86_64-comm.tar`

   - Evaluation:

     `tectia-server-<version>-linux-ubuntu-x86_64-upgrd-eval.tar`

   In the package names, *<version>* is the current product release (for example, `6.6.5.123-1`).

3. Unpack the downloaded `tar` package.

4. Select the installation packages (in this example, we install Tectia Server only). Two packages are always required: one for the common components of Tectia Client and Server, and one for the specific components of Tectia Server.

   `ssh-tectia-common-<version>_linux-x86_64.deb`

```
ssh-tectia-server-<version>-linux-x86_64.deb
```

5. Install the packages with root privileges:

```
# dpkg -i ssh-tectia-common-<version>_linux-x86_64.deb
# dpkg -i ssh-tectia-server-<version>_linux-x86_64.deb
```

Note

If you have already installed Tectia Client, you don't need to install the `common` file again.

The server host key is generated during the initial installation. The key generation may take several minutes on slow machines.

6. Copy the license file to the `/etc/ssh2/licenses` directory. (*This is not necessary in "third-digit" maintenance updates.*)

   If this is the initial installation of Tectia, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start Tectia Server manually after copying the license file.

7. The installation should (re)start Tectia Server automatically.

   If Tectia Server does not start (for example because of a missing license or because some other secure shell software is running on port 22), you can start it after correcting the problem by issuing the command:

```
# ssh-server-ctl start
```

2.2.5 Installing on Solaris

The downloaded installation package contains the compressed installation files.

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

Tectia Server includes support for Zones on Solaris 10 and 11. The Tectia software can be installed into the global and local zones. When the Tectia software is installed into the global zone, it becomes automatically installed also into the existing local zones. However, Tectia Server needs to be separately installed into local zones added later into the system.

In case you are installing Tectia Server 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/tectia/bin` to the path settings.

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

To install Tectia Server on Solaris, follow the instructions below:

1. Unpack the downloaded `tar` package.

2. Make sure no other software is using port 22 (Tectia Server default listen port). Stop any competing server software or change their listen port.

3. Select the installation package according to your Solaris version.

   When installing on Solaris version 10 running on the SPARC architecture, use the following packages:

   ```
   ssh-tectia-common-<version>-solaris-10-sparc.pkg.Z
   ssh-tectia-server-<version>-solaris-10-sparc.pkg.Z
   ```

   When installing on Solaris version 11 running on the SPARC architecture, use the following packages:

   ```
   ssh-tectia-common-<version>-solaris-11-sparc.pkg.Z
   ssh-tectia-server-<version>-solaris-11-sparc.pkg.Z
   ```

   When installing on Solaris version 10 or 11 running on the x86-64 architecture, use the following packages:

   ```
   ssh-tectia-common-<version>-solaris-<solaris-version>-x86_64.pkg.Z
   ssh-tectia-server-<version>-solaris-<solaris-version>-x86_64.pkg.Z
   ```

   In the commands, *<version>* indicates the product release version and the current build number (for example, `6.6.5.123`). *<solaris-version>* refers to the Solaris version number (10 or 11), in case of installing on x86-64 architecture.

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

   ```
   $ uncompress ssh-tectia-common-<version>-solaris-10-x86_64.pkg.Z
   $ uncompress ssh-tectia-server-<version>-solaris-10-x86_64.pkg.Z
   ```

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

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

   The server host key is generated during the installation. The key generation may take several minutes on slow machines.

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

   If this is the initial installation of Tectia Server, the directory does not yet exist. You can either create it manually or copy the license after the installation. In the latter case, you have to start the server manually after copying the license file.

7. The installation should (re)start the server automatically.

Note

If the server does not start (for example because of a missing license or because some other secure shell software is running on port 22), you can start it after correcting the problem by issuing the command:

```
# /etc/init.d/ssh-server-g3  start
```

Tip

On Solaris, it is recommended that you raise the maximum open files limit. The default limit for open files per process is set to 256, but it is too low for Tectia Server that will receive lots of connections. The servant may run out of file descriptors causing the connections to fail.

How much the maximum open files limit must be raised, depends on the system and the number of servants running; 8192 should be sufficient in most cases.

To set the maximum open files limit to 8192, before starting **ssh-server-g3**, run this command in shell:

```
# ulimit -n 8192
```

The default limit set for open files varies between operating system versions. Refer to the instructions of your operating system for more information.

In case you want to use the BSM to record Secure Shell log-in and log-out events, see also Section 6.2.3.

2.2.6 Installing on Windows

The Windows installation package is provided in the MSI (Windows Installer) format for Microsoft Windows versions running on the 64-bit (x86-64) platform architecture. Tectia Server installation packages can be used to install also Tectia Client.

The installation package is a zip file containing the Tectia Client/Server license files and the executable Windows Installer (MSI) packages.

You must have administrator rights to install Tectia Client/Server on Windows.

For Tectia Client/Server to be fully functional after installation, you must restart the computer.

Note

If you do not restart the computer after installing Tectia Server, the server will run with the following limitations in the authentication of local users and domain users from one-way trusted domains:

- Public-key authentication will not work.
- Certificate authentication, keyboard-interactive submethods RADIUS and RSA SecurID, and host-based authentication will only work if the password cache (see Section 4.1.5) is enabled and the user's password is stored in the cache.
- Authentication selectors of type **User group** (`user-group`) and **Administrator** (`user-privileged`) will not work. (For more information on selectors, see the section called "Editing Selectors".)

Tectia Server will write warning messages into the Windows Event Log. Use the Windows Event Viewer to examine the log contents (On the **Tectia Server Configuration** tool's **Tectia Server** page, click the **View Event Log** button.

Note

Tectia Server cannot be installed on file systems that do not support permissions (for example, FAT16 or FAT32). The hard disk partition where Tectia Server is installed must use the NTFS file system.

The installation is carried out by a standard installation wizard. The wizard will prompt you for information and will copy the program files, install the services, and generate the host key pair for the server.

To install Tectia Server and (optionally) Tectia Client on Windows, follow the instructions below:

1. Make sure no other software is using port 22 (Tectia Server default listen port). Stop any competing server software or change their listen port. Also make sure the firewall is open for incoming connections to TCP port 22.
2. Extract the contents of the installation zip file to any temporary location.
3. Locate the correct Windows Installer file `ssh-tectia-server-<version>-windows-<platform>.msi`, where:
   - `<version>` shows the Tectia Client/Server release version and build number, for example `6.6.5.123`.
   - `<platform>` shows the platform architecture `x86_64` for 64-bit Windows versions.
4. Double-click the installation file, and the installation wizard will start.

Note

The license files will be imported automatically when you extract the contents of the `.zip` package before running the `.msi` installer.

If you run the `.msi` installer directly from the `.zip` package, you need to manually import the license files (`sts66.dat` for Tectia Server and `stc66.dat` for Tectia Client) after completing the installation. The installation wizard will show an error message about missing license files, and when you attempt to start Tectia Client/Server, you are prompted to import the license(s) manually to the license directory:

- `"C:\Program Files (x86)\SSH Communications Security\SSH Tectia\SSH Tectia AUX\licenses"` on 64-bit Windows versions

On Windows 10, Tectia packages downloaded via browser may trigger a *Windows protected your PC* warning. In such cases, proceed with the installation by clicking **More info** and **Run anyway**.

5. Follow the wizard through the installation steps and fill in information as requested.

   The installation wizard will display options **Typical**, **Custom** and **Complete**.

   If you do not want to install both Tectia Server and Client, select **Custom** and choose which product components you wish to install.

   The server host key is generated during the installation.

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

7. Fresh installation always requires restarting the computer. In case you were performing an upgrade, a restart is not necessarily required.

8. Restart the computer.

   Tectia Server will start automatically every time the computer is started, and it stays running in the background. Tectia Server displays no icons on the desktop, but you can see it listed in the Windows **Start → Programs** menu.

   In case the server does not (re)start automatically, you can start it manually according to the instructions given in Section 3.1.3.

Silent Installation

Tectia Server can also be installed silently on a server host. 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, Tectia Server is installed with the default settings and without any additional features.

Note

After Tectia Server has been installed, it is automatically restarted.

The following command can be used to install Tectia Server silently:

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

In the command:

- `<version>` shows the current version of Tectia Server, for example `6.6.5.123`.
- `<platform>` shows the platform architecture `x86_64` for 64-bit Windows versions.
- `<path>` is the path to the desired installation directory. If the `INSTALLDIR` variable is omitted, Tectia Server is installed to the default location.

The above command installs all features available in the Tectia Server installer, including Tectia Client. If you wish to install only Tectia Server, use the ADDLOCAL property as follows:

```
msiexec /q /i ssh-tectia-server-<version>-windows-<platform>.msi ADDLOCAL=tectia_server \
INSTALLDIR="<path>"
```

It is also possible to use the Tectia Server installer to install only Tectia Client:

```
msiexec /q /i ssh-tectia-server-<version>-windows-<platform>.msi ADDLOCAL=tectia_client \
INSTALLDIR="<path>"
```

2.3 Removing the Tectia Server Software

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

Note

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

2.3.1 Removing from AIX

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

1. Stop Tectia Server by using the System Resource Controller (SRC) of the operating system:

   ```
   # stopsrc -s ssh-tectia-server
   ```

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

   ```
   # installp -u SSHTectia.Server
   ```

3. If you want to remove also the components that are common with Tectia Client, give the following command:

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

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

2.3.2 Removing from HP-UX

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

1. Stop Tectia Server with the following command:

   ```
   # /sbin/init.d/ssh-server-g3 stop
   ```

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

   ```
   # swremove SSHG3server
   ```

3. If you want to remove also the components that are common with Tectia Client, give the following command:

```
# swremove SSHG3common
```

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

2.3.3 Removing from Linux

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

1. Log in as root user.
2. Stop Tectia Server.
   - Using Tectia Server control utility:

     ```
     # ssh-server-ctl stop
     ```

   - Or on Linux with systemd:

     ```
     # systemctl stop ssh-server-g3
     ```

   - Or on Linux without systemd:

     ```
     # /etc/init.d/ssh-server-g3 stop
     ```

3. Remove the installation in Linux by giving the following command:

   ```
   # rpm -e ssh-tectia-server
   ```

   Or on Debian Linux:

   ```
   # dpkg -P ssh-tectia-server
   ```

4. If you want to remove also the components that are common with Tectia Client, give the following command:

   ```
   # rpm -e ssh-tectia-common
   ```

   Or on Debian Linux:

   ```
   # dpkg -P ssh-tectia-common
   ```

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

2.3.4 Removing from Solaris

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

1. Stop Tectia Server with the following command:

```
# /etc/init.d/ssh-server-g3 stop
```

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

```
# pkgrm SSHG3srvr
```

Note

Tectia Server needs to be uninstalled separately from each local zone, if it has been installed to all zones by installing into the global zone.

3. If you want to remove also the components that are common with Tectia Client, give the following command:

```
# pkgrm SSHG3cmmn
```

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

2.3.5 Removing from Windows

To remove Tectia Server from a Windows environment, follow the instructions below:

1. From the Windows **Start** menu, open the **Control Panel** and click **Programs and Features**.
2. From the program list, select **Tectia Server** and click **Uninstall**.

Note

If you have installed Tectia Client together with Tectia Server, uninstalling Tectia Server will also remove Tectia Client.

3. Click **Yes** to confirm.

Note

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

2.4 Files Related to Tectia Server

This section lists the default locations where you will find the installed executables, configuration files, key files, the license file, and the user-specific configuration files after the installation phase.

The required file permissions (read and write rights) are also listed and marked:

MUST if security is compromised if these permissions are incorrect.

SHOULD if security is not be compromised, but incorrect permissions would give away information.

2.4.1 File Locations and Permissions on Unix

On Unix platforms, the Tectia Server files are located in the following directories and the named file permissions are required for them:

- `/etc/ssh2`

  Writable to *root* (must). Readable to *world*. The `/etc/ssh2` directory is created with the correct permissions during installation.

  - `/etc/ssh2/ssh-server-config.xml`: the server configuration file (see ssh-server-config(5))

    Writable to *root* (must). Readable to *world*.

  - `/etc/ssh2/ssh-server-config-default.xml`: a sample file that shows the hardcoded system defaults of the server configuration

  - `/etc/ssh2/ssh-server-config-example.xml`: a sample file with useful examples for the server configuration

  - `/opt/tectia/share/auxdata/ssh-server-ng`: the server configuration file DTD directory

  - `/etc/ssh2/hostkey`: the default server host private key file

    Writable to *root* (must). Readable to *root* (must).

  - `/etc/ssh2/hostkey.pub`: the default server host public key file

    Writable to *root* (should). Readable to *world*.

  - `/etc/ssh2/hostkey.pass`: the default server host key passphrase file if the host private key has been encrypted.

    Writable to *root* (must). Readable to *root* (must).

  - `/etc/ssh2/licenses`: the license file directory (see Section 2.1.3)

  - `/etc/ssh2/trusted_hosts`: the directory for host public keys that are trusted for host-based authentication (see Section 5.8)

    Writable to *root* (must). Readable to *root* (should).

- `/var/opt/tectia/random_seed`: the seed file for the random number generator

  Writable to *root* (must). Readable to *root* (must). Set the permissions read/writable to *root* at each update.

- `/opt/tectia/sbin`: the system binaries such as **ssh-server-g3** and its control utility **ssh-server-ctl**

- `/opt/tectia/bin`: the user binaries such as **ssh-keygen-g3**
- `/opt/tectia/man`: Tectia Server man pages
- `/opt/tectia/libexec`: library binaries
- `/opt/tectia/lib/sshsecsh`: library binaries

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

Writable to *user* (must). Readable to *user* (should). The permission checking can be changed with configuration setting `<auth-file-modes mask-bits="XXX"/>`.

In the `$HOME/.ssh2` directory:

- `$HOME/.ssh2/authorized_keys`: the default directory for user public keys that are authorized for login
- `$HOME/.ssh2/authorization`: (*optional*) the default authorization file for user public keys

2.4.2 File Locations on Windows

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

- `"C:\Program Files (x86)\SSH Communications Security\SSH Tectia"` on 64-bit Windows versions

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

- `"<INSTALLDIR>\SSH Tectia Server"`: system binaries such as **ssh-server-g3.exe**
  - `"<INSTALLDIR>\SSH Tectia Server\ssh-server-ctl.exe"`: server control utility command-line tool

    ### Note

    To use the server control utility, the *Windows PowerShell* or `cmd.exe` has to be started with *Run as Administrator* and the control utility executed from its install directory `"<INSTALLDIR>\SSH Tectia Server\"`, for example **.\ssh-server-ctl.exe status**

  - `"<INSTALLDIR>\SSH Tectia Server\ssh-server-config.xml"`: server configuration file (see ssh-server-config(5))

    ### Note

    For the server (and its configuration tool) to start, the configuration file must have correct permissions. Make sure that the owner of the file is a member of the *Administrators* group, only *Administrators* and *SYSTEM* may have full control of the file, *Users* are not allowed to modify the file, and other accounts do not have access to the file.

- "`<INSTALLDIR>\SSH Tectia Server\ssh-server-config-default.xml`": sample file that shows the hardcoded system defaults of the server configuration
- "`<INSTALLDIR>\SSH Tectia Server\ssh-server-config-example.xml`": sample file that shows useful examples for the server configuration
- "`<INSTALLDIR>\SSH Tectia Server\hostkey`": default server host private key file

  Full permissions allowed only for *Administrators* group and the *SYSTEM* account.
- "`<INSTALLDIR>\SSH Tectia Server\hostkey.pub`": default server host public key file

  Full permissions allowed only for *Administrators* group and the *SYSTEM* account. Read permissions for *Users* group.
- `<INSTALLDIR>\hostkey.pass`: the default server host key passphrase file if the host private key has been encrypted.

  Full permissions allowed only for *Administrators* group and the *SYSTEM* account.
- "`<INSTALLDIR>\SSH Tectia Server\random_seed`": the seed file for the random number generator
- "`<INSTALLDIR>\SSH Tectia Server\trusted_hosts`": directory for host public keys that are trusted for host-based authentication (see Section 5.8)

- "`<INSTALLDIR>\SSH Tectia AUX`": auxiliary binaries such as **ssh-keygen-g3.exe**
- "`<INSTALLDIR>\SSH Tectia AUX\ssh-server-ng`": server configuration file DTD directory
- "`<INSTALLDIR>\SSH Tectia AUX\licenses`": license file directory (see Section 2.1.3)

Note

Users that log on to SSH server require **Read & execute** permissions for the following files in the folder `<INSTALLDIR>\SSH Tectia AUX`:

- `i18n_icu.dll`
- `icudt40.dll`
- `icuuc40.dll`

In addition, a system library file is copied to a Windows directory:

- "`C:\WINDOWS\system32\sshdap.dll`": library file for SSH-specific domain authentication package (DAP)

Figure 2.2 shows the Tectia directory structure when also Tectia Client has been installed on the same machine.

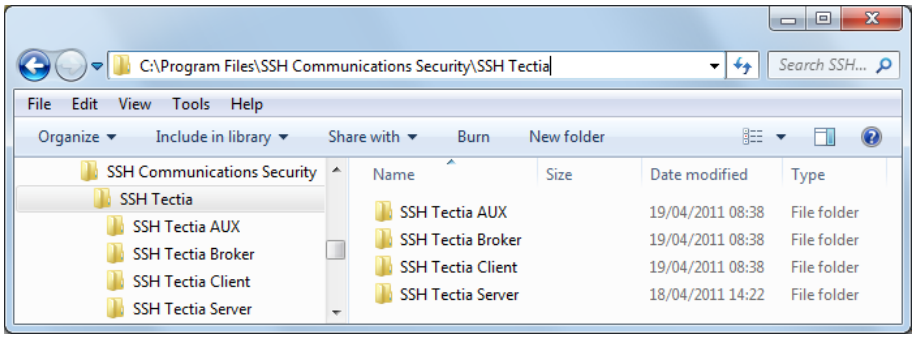


Figure 2.2. The Tectia directory structure on Windows

The user-specific configurations are stored in each user's own directory:

- `%USERPROFILE%\.ssh2\authorized_keys\`: the default directory for user public keys that are authorized for login
- `%USERPROFILE%\.ssh2\authorization`: (*optional*) the default authorization file for user public keys.

2.4.3 Registry Keys on Windows

On Windows, the Tectia Server installation creates the following registry keys:

- `HKCU\SOFTWARE\SSH Communications Security\SSH Tectia\KeyPaths`
- `HKLM\SOFTWARE\SSH Communications Security\SSH Tectia Server`
- `HKLM\SOFTWARE\Wow6432Node\SSH Communications Security\SSH Tectia` (on x64 architecture, only)
- `HKLM\SOFTWARE\Wow6432Node\SSH Communications Security\SSH Tectia Server` (on x64 architecture, only)
- `HKLM\SYSTEM\CurrentControlSet\Services\SSHTectiaServer`
- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia SFT Server`
- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia Server`
- `HKLM\SYSTEM\CurrentControlSet\Control\Lsa`
- `HKLM\SYSTEM\CurrentControlSet\Control\Session Manager`

Chapter 3 Getting Started

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

The default configuration of Tectia Server is usable in most environments. For advice on configuring the servers, see Chapter 6, Chapter 7, and Chapter 8.

Before you proceed, see general Tectia product information in *Tectia Client/Server Product Description* (a separate document).

3.1 Starting and Stopping the Server

Tectia Server is started automatically after installation and at boot time. Tectia Server uses a distributed architecture where the master server process launches several servant processes that handle the actual connections. The number of servants per master server process is configurable within certain limits. See Section 4.1.2.

3.1.1 Starting and Stopping on AIX

If the server needs to be started or stopped manually on AIX platforms, use the System Resource Controller (SRC) of the operating system.

To start Tectia Server, enter command:

```
startsrc -s ssh-tectia-server
```

To stop Tectia Server, enter command:

```
stopsrc -s ssh-tectia-server
```

On AIX, using **startsrc** starts two ssh-server-g3 processes. One process is so-called service launcher that interfaces with the SRC and the actual SSH server process. By using a separate service launcher, the SRC is able to start a new server process in the case that old server process has been stopped but it is still serving open connections.

You can also use **ssh-server-ctl** (Tectia Server control utility) to start and stop the server, but it will actually call the AIX system resource controller.

To start Tectia Server with the control utility, enter:

```
ssh-server-ctl start
```

To stop Tectia Server with the control utility, enter:

```
ssh-server-ctl stop
```

3.1.2 Starting and Stopping on Other Unix Platforms

To manually start, stop, or restart the server:

- On Linux with systemd:

  ```
  # systemctl [command] ssh-server-g3
  ```

- On Solaris, and Linux without systemd:

  ```
  # /etc/init.d/ssh-server-g3 [command]
  ```

- On HP-UX:

  ```
  # /sbin/init.d/ssh-server-g3 [command]
  ```

The `command` can be:

start

Start the server.

stop

Stop the server. Existing connections stay open until closed from the client side.

restart

Start a new server process. Existing connections stay open using the old server process. The old process is closed after the last old connection is closed from the client side.

reload

Reload the configuration file. Existing connections stay open.

3.1.3 Starting and Stopping on Windows

There are several ways to start and stop Tectia Server on Windows. Select a suitable method from below.

Using the Services Console:

1. From the **Start** menu, open the Windows **Control Panel** and double-click **Administrative Tools**.

2. Double-click **Services**. The **Services** console opens.

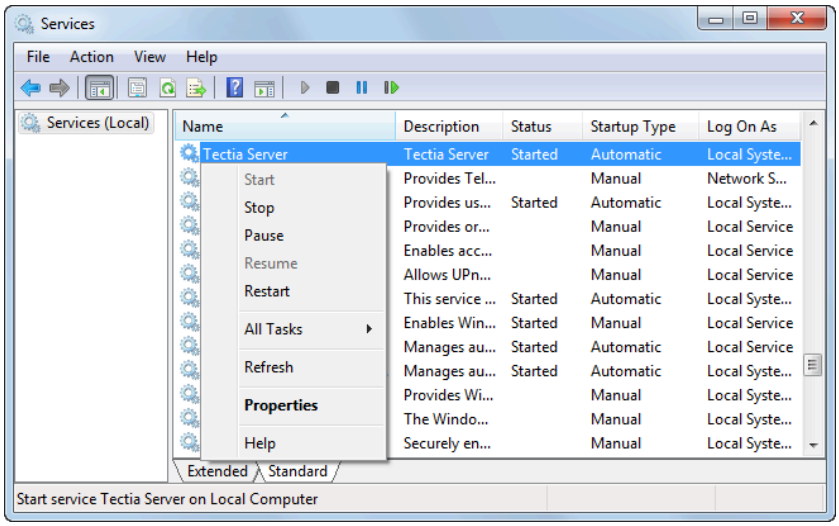


Figure 3.1. Starting and stopping Tectia Server from the Windows Services console

3. From the list, right-click on Tectia Server. From the shortcut menu, you can now **Start**, **Stop**, **Pause**, **Resume**, or **Restart** the server.

   If the server is paused, the existing connections will stay open but the server will not accept new connections.

Using the Tectia Server Configuration GUI:

1. Open the Tectia Server Configuration GUI.

2. Click the **Start Server** button or **Stop Server** button. The button name and the function changes according to the state of the Server.

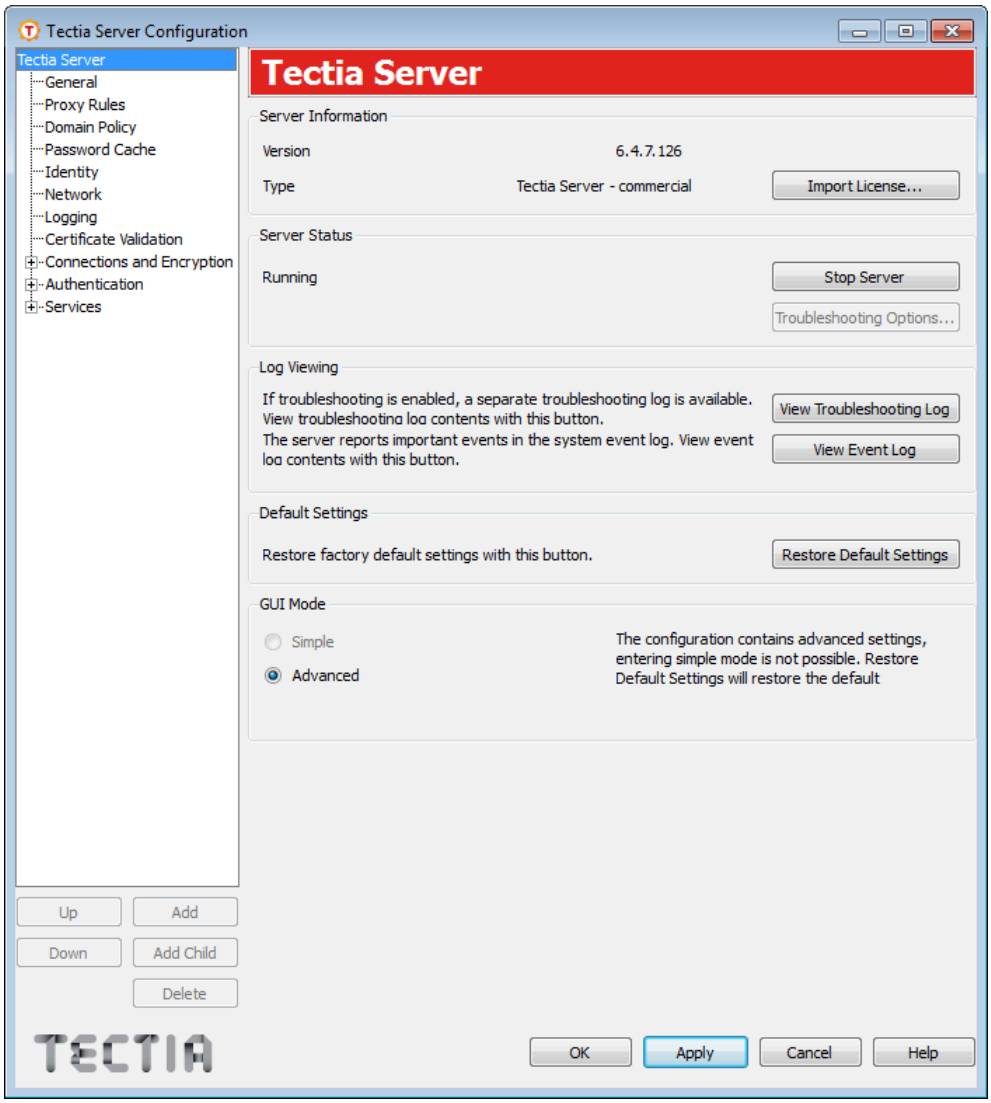


Figure 3.2. Using Tectia Server Configuration GUI to start and stop the Server

Chapter 4 Configuring Tectia Server

Tectia Server uses an XML-based configuration file `ssh-server-config.xml` that allows flexible implementation of real-life enterprise security policies.

The configuration file can be used to define settings with values that are different from the factory-set default values. When the configuration file has been created, the values of elements included in the file will override the default values of those elements. Any elements not included in the configuration file will use the hard-coded default values.

You can view the default values in the `ssh-server-config-default.xml` file that is stored in `/etc/ssh2/` on Unix and in `<INSTALLDIR>\SSH Tectia Server\` on Windows. The default configuration file is not read by Tectia Server, but it shows the hardcoded system defaults.

Tectia Server also includes an example file `ssh-server-config-example.xml` that contains a useful example configuration with explanations of the options. The example file is located in the same directory as the default configuration file.

On Windows, you can use the **Tectia Server Configuration** tool to edit the configuration (see Section 4.1). The `ssh-server-config.xml` configuration file can also be edited with an XML editor or ASCII text editor directly in XML format (see Section 4.2).

After editing the configuration file, in most cases it is enough to reconfigure the server, but changing the listener ports or the FIPS-mode settings requires restarting the server. On Windows, reconfiguration happens when you click **Apply** or **OK**. On Unix, to make Tectia Server re-read its configuration, you can use the ssh-server-ctl(8). For instructions on restarting the server, see Section 3.1.

4.1 Tectia Server Configuration Tool

The easiest way to configure the server is to use the **Tectia Server Configuration** tool. Start the program by clicking the **Tectia Server Configuration** icon in the **Tectia Server** program group (or by running the `ssh-server-gui.exe` program located in the installation directory on Windows).

The **Tectia Server Configuration** tool displays the settings in a tree structure. Select the desired configuration page by clicking the list displayed on the left. The order of settings in the **Tectia Server Configuration** tool follows the same logic that is used in setting up a Secure Shell connection:

1. The settings related to the server's own configuration are made with the following configuration pages:

   - **Tectia Server** (see Section 4.1.1)
   - **General** (see Section 4.1.2)
   - **Proxy Rules** (see Section 4.1.3)
   - **Domain Policy** (see Section 4.1.4)
   - **Password Cache** (see Section 4.1.5)
   - **Identity** (see Section 4.1.6)
   - **Network** (see Section 4.1.7)
   - **Logging** (see Section 4.1.8)
   - **Certificate Validation** (see Section 4.1.9).

2. After the client has initiated a connection to the server, the server checks whether connections from the client address are allowed. The client and server perform key exchange where the server authenticates itself to the client, and ciphers, KEXs and MACs are selected for the connection. The related configuration settings are made with the **Connections and Encryption** configuration page (see Section 4.1.11).

3. The server requests the user to authenticate itself to the server. The server may offer a selection of authentication methods or require several authentication methods to be passed in succession. The configuration settings related to authentication are made with the **Authentication** configuration page (see Section 4.1.12).

4. The server determines the services the client is allowed to use. The related configuration settings are made with the **Services** configuration page (see Section 4.1.13).

For troubleshooting instructions, see also Chapter 9.

4.1.1 Tectia Server

The **Tectia Server** page allows you to start and stop the server, adjust troubleshooting settings, view the event log, restore the settings to their default values, and select the GUI mode (simple or advanced).

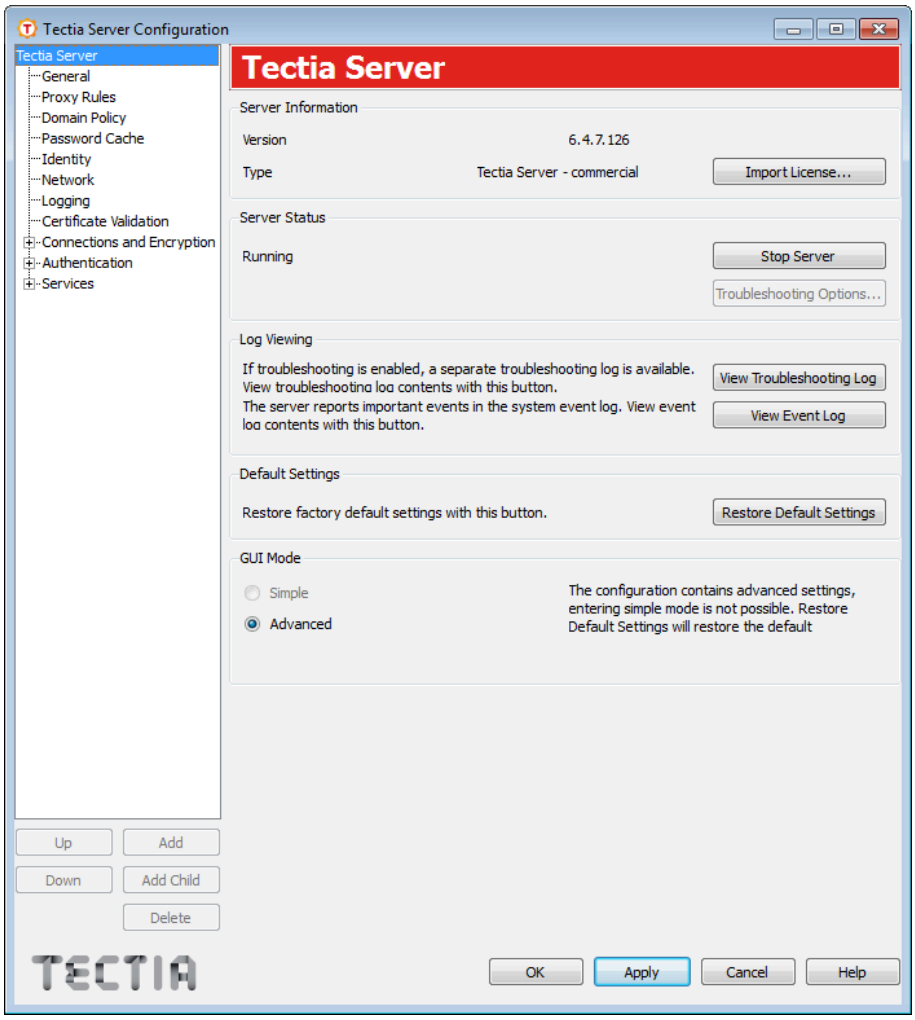


Figure 4.1. Tectia Server Configuration - Tectia Server page

Server Information

The server version and the license type (commercial or evaluation) are shown at the top of the page.

If a valid license file cannot be found, the text "**License not available**" is shown in place of the server and license type. The server will not start without a license file. For more information, see Section 2.1.3.

Server Status

The server status is displayed on the left. The status can be either **Stopped**, **Starting**, **Running**, **Stopping**, **Paused**, **Pausing**, **Continuing**, or **Failure**. To start or stop the server, click the **Start Server/Stop Server** button.

When the server is stopped, clicking the **Troubleshooting Options** button opens a dialog box where you can set the server to run in troubleshooting mode.

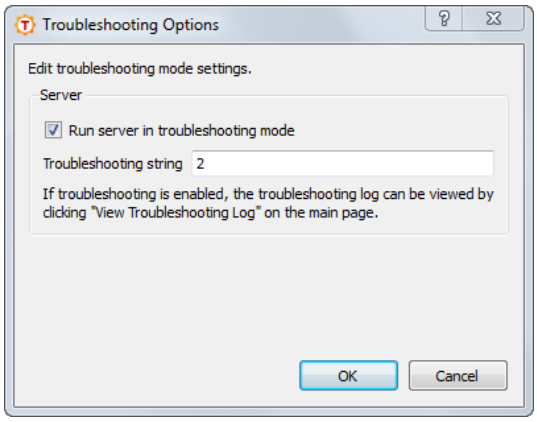


Figure 4.2. Editing troubleshooting options

To run the server in troubleshooting mode, select the check box and enter a troubleshooting string. The string can be a number from 1 to 99 (the default is *2*). The higher the number, the more detailed troubleshooting output is generated.

When the server is started again, the troubleshooting log can be viewed by clicking **View Troubleshooting Log**.

Log Viewing

When the server is running in troubleshooting mode, the log can be viewed by clicking **View Troubleshooting Log**. This opens a separate window where the log is displayed. The log is displayed from the moment the window is opened.

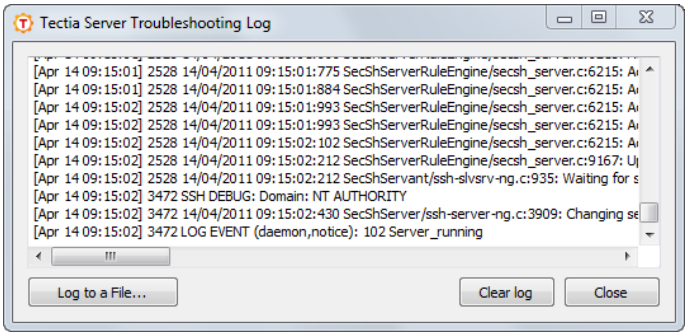


Figure 4.3. Viewing troubleshooting log

In the log window, you can select the text normally and copy it to clipboard. If you want the whole log saved to a file, click **Log to a File**, and define the file name and location. To clear the log window, click **Clear log**. To close the window, click **Close**.

Note that closing the window does not affect the mode the server is running in. Reopening the window will again display the log from the moment of opening.

Important events are logged in the system event log. Click the **View Event Log** button to launch the **Event Viewer** program that allows you to examine the log contents. For more information, see Section 6.2.

Default Settings

To discard any changes you have made to the configuration settings and restore the factory default values, click the **Restore Default Settings** button and click **OK** in the confirmation dialog.

Note that the settings will not revert back to the previously saved values, but to the initial values that are built into the program.

GUI Mode

The **Tectia Server Configuration** tool can be run in two GUI modes: **Simple** and **Advanced**.

In the simple GUI mode, you cannot add multiple rules under the **Connections and Encryption**, **Authentication**, and **Services** pages, and the **Selector** tabs are not available. The same connection, authentication, and service rules are applied to all users. If you have already defined selectors or added several rules, you cannot enter simple mode. Clicking **Restore Default Settings** will restore the default configuration and allow entering simple mode again.

In the advanced mode, all settings in the GUI are available. You can add several rules under **Connections and Encryption**, **Authentication**, and **Services** pages and define selectors for them. See Section 4.1.11, Section 4.1.12, and Section 4.1.13,.

4.1.2 General

The **General** page contains the general server settings, for example, the maximum number of connections and processes, settings for load control, FIPS mode, and banner message.

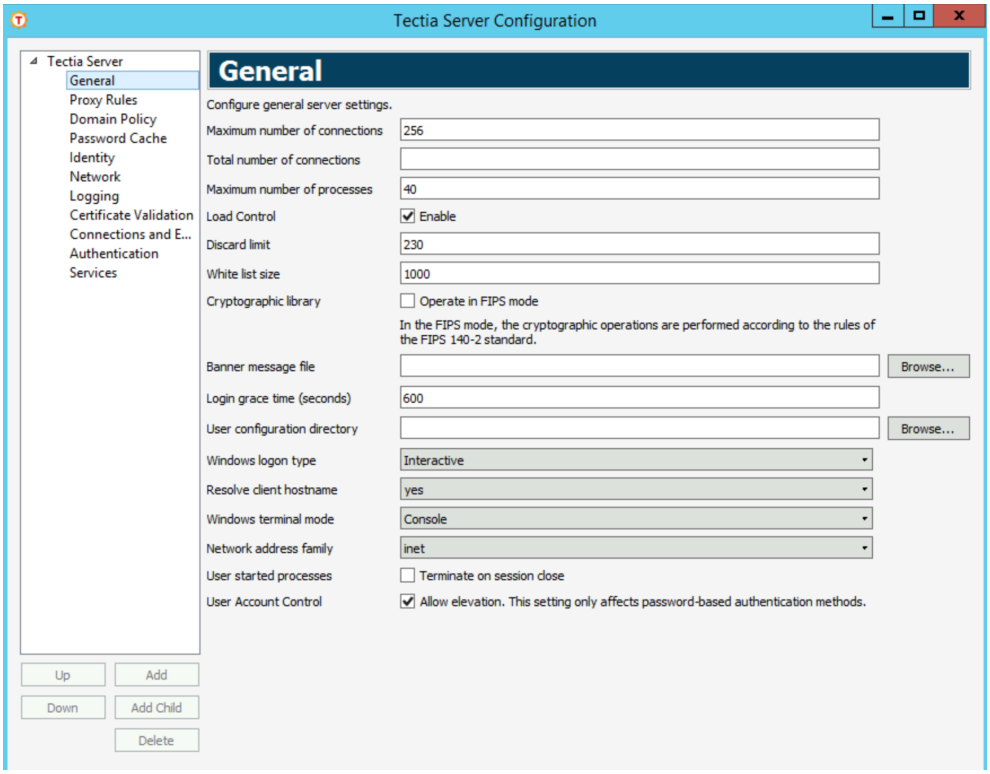


Figure 4.4. Tectia Server Configuration - General page

Maximum number of connections / Total number of connections / Maximum number of processes

Tectia Server uses a distributed architecture where the master server process launches several servant processes that handle the actual client connections. The server's total number of connections is the number of connections multiplied by the number of processes.

Limiting the maximum number of connections is useful in systems where system overload may be caused by a high load in the server program when opening new connections.

Maximum number of connections defines the maximum number of client connections allowed per servant. The default (and recommended) value is 256.

Total number of connections defines the maximum number of connections that a servant will handle before the server should start a new servant in its place. The allowed value range is 1-4,000,000,000. If no value is given (default), the servant-lifetime functionality will be disabled and the servants are never retired. This corresponds to the `servant-lifetime` element in the server configuration file (see **servant-lifetime**).

Maximum number of processes defines the maximum number of servant processes the master server will launch. The value range is 1 to 2048. The default (and recommended) value is 40.

The maximum number of connections a server can handle depends on system resources, including the maximum number of open file descriptors, the maximum number of processes available to a single user, the maximum number of available PIDs, and the amount of memory available.

Load Control / Discard limit / White list size

Load Control defines settings for keeping Tectia Server working when the load is high, that is, the number of current connections is near the maximum allowed number of connections. High load might be caused by a connection flood denial-of-service attack that tries to make the server unavailable to its intended users by using so much of its resources that normal service is disrupted. Load control is enabled by default. To disable load control, clear the **Enable** check box.

Note

If **Maximum number of connections** is set to 1, load control will be disabled.

Load control is implemented by keeping a "white list" of the IP addresses of connections that have had a successful authentication. When Tectia Server starts, the white list is empty. When the server's load is high, connections from IP addresses that are not on the white list (that is, connections that have not recently had a successful authentication) are discarded.

When the number of a servant's concurrent connections is not higher than the value of **Discard limit**, the servant accepts connections from any IP address. When the number of a servant's concurrent connections exceeds the **Discard limit**, only connections from IP addresses that are on the server's white list are accepted. If existing servants cannot accept any more connections, but the **Maximum number of processes** (that is, the maximum number of servant processes the master server will launch) limit has not been reached, the server launches a new servant process which will accept new connections.

The allowed value range for **Discard limit** is 1 to **Maximum number of connections** - 1. The default value is 90 percent of the value of **Maximum number of connections**.

White list size specifies the number of IP addresses on the server's white list. The allowed value range is 1 to 10000. The default value is 1000.

Cryptographic library

Tectia Client, ConnectSecure, and Server can be operated in FIPS mode, using a version of the cryptographic library that has been certified according to the Federal Information Processing Standard (FIPS) 140-2.

The full OpenSSL cryptographic library is distributed with Tectia Server. This OpenSSL FIPS-certified cryptographic library is used to provide the classes of functions listed in the following tables.

The functions from the OpenSSL 3.0.8 7 Feb 2023 (FIPS provider: 3.0.8) used on Linux, Windows, and Solaris are listed in Table 4.1.

Table 4.1. APIs used from the OpenSSL cryptographic library version 3.0.8

API	Description	Functions from OpenSSL
Random numbers	AES/CTR DRBG based on NIST SP800-90A is used from the OpenSSL library.	RAND_bytes, RAND_add
Ciphers	aes-ecb, aes-cbc, aes-ofb, aes-ctx, aes-gcm 3des-(ecb,cbc,cfb,ofb)	EVP_CIPHER_CTX_*, EVP_Cipher*
Math library	Bignum math library used by OpenSSL.	BN_*
Diffie Hellman	DH, ECDH, curve25519, curve448	EVP_PKEY_*, DH_*
Hash functions	Variants: sha1[verify only], sha224, sha256, sha384, sha512	EVP_MD_*, EVP_sha*, EVP_Digest*
Public Key	Variants: RSA, DSA, ECDSA, Ed25519	EVP_PKEY_*, i2d_DSA_SIG, d2i_DSA_SIG, i2d_ECDSA_SIG, d2i_ECDSA_SIG, EVP_MD_*, ECDSA_SIG_*, DSA_SIG_*, EC_GROUP_*, EC_POINT_*
Misc		ERR_error_string_n, ERR_get_error, OpenSSL_version OSSL_PARAM_*, OSSL_PROVIDER_*, CRYPTO_free, CONF_modules_load_file_ex, EVP_default_properties_enable_fips

No certificate functions are used from the OpenSSL library. Tectia provides its own certificate libraries.

Select the **Operate in FIPS Mode** check box to use the FIPS-certified version of the SSH cryptographic library. Clear the check box to use the standard (default) SSH cryptographic library.

Note

Tectia Server has to be restarted after changing the FIPS-mode setting. Extra checks are done when starting Tectia Server and Connection Broker in the FIPS mode due to the OpenSSL FIPS crypto library health check. This will lead to a noticeable delay in the start of the process on slow machines.

Banner message file

To define a banner message file, click the **Browse** button on the right-hand side of the text field. The **Select File** dialog appears, allowing you to specify the desired file. You can also type the path and file name directly into the text field.

The message file is sent to the client before authentication. Note, however, that the client is not obliged to show this message.

Login grace time

Specify a time after which the server disconnects if the user has not successfully logged in. If the value is set to `0`, there is no time limit. The default is `600` seconds.

User configuration directory

Specify a path to a directory from where Tectia Server looks for user-specific authorized public keys, if they are not stored to the default location. With this setting the administrator can control options that are usually controlled by the user. If no setting is given, the default setting will be used.

The default setting is `%D/.ssh2`, which expands to `%USERPROFILE%\.ssh2` (usually "`C:\Documents and Settings\<username>\.ssh2`").

Enter the path as a pattern string which will be expanded by Tectia Server. The following pattern strings can be used:

- `%D` or `%homedir%` is the user's home directory
- `%U` or `%username%` is the user's login name

  For Windows domain users:

  - `%U` is expanded to `domain.username`
  - `%username%` is expanded to `domain\username`

  For local server machine users:

  - `%U` is expanded to `username`
  - `%username%` is expanded to `username` (without the domain prefix)
- `%username-without-domain%` is the user's login name without the domain part.

- `%installdir%` is the installation directory.

Note

The **User configuration directory** setting will be read only if the **Authentication** view does NOT have anything set in the following settings under **Public-Key Authentication**:

- **Authorization file**
- **Authorized-keys directory**
- **OpenSSH authorized-keys file**

For reference, see the section called “Parameters”

Windows logon type

Specify what kind of user logon methods for the local host are accepted by Tectia Server. The defined logon type affects password authentication. Select a suitable value from the drop-down list: **Batch**, **Interactive**, **Network**, or **Network-Cleartext**. The default value is **Interactive**. Note that this setting only affects password-based authentication methods.

For example, to enable accounts that do not have the access right to log on locally, select **Network**.

For information on the attribute values, refer to Microsoft documentation on Windows logon types.

Resolve client hostname

Define whether Tectia Server should try to resolve the client host name from the client IP address during connection setup. By default, **yes** is selected and DNS lookups are used to resolve the client host name at connection time.

If you select **no**, client host name resolution is not attempted, but the IP address is used as the returned client host name. This is useful when you know that the DNS cannot be reached, and the query would cause just additional delay in logging in.

Note

This attribute does not affect the resolution of TCP tunnel endpoints and Tectia Server will try to resolve the client host name when creating a TCP tunnel.

Windows terminal mode

Define the mode of operation of a terminal session on the server side. The available values are **Console**, **Console-UTF8** and **Stream**.

If set to **Console** (default), the server reads the screen buffer in a loop and detects modifications based on current cursor location. In **Console-UTF8** mode the console is handed in Unicode mode and data is converted to and from UTF-8 for the client. If set to **Stream**, the server reads the stdout and stderr of **cmd.exe** as a stream of data, while providing basic facilities for command-line editing.

Network address family

Define the address family Tectia Server will use for incoming connections.

If set to **inet** (default), the server will accept only IPv4 incoming connections. If set to **inet6**, the server will accept only IPv6 incoming connections. If set to **Any**, the server will accept both IPv4 and IPv6 incoming connections, will resolve addresses of both families, and opens both IPv4 and IPv6 listeners for remote port forwarding.

User started processes

Select the **Terminate on session close** check box to have all processes started by the user on the SSH terminal session terminated when the user logs off from the session. By default this is not enabled.

User Access Control

If **Allow elevation** is selected, users logging in with password authentication may retain any admin privileges associated with their accounts.

4.1.3 Proxy Rules

On the **Proxy Rules** page, you can define rules for HTTP or SOCKS proxy servers that Tectia Server uses when a client requests local port forwarding (local tunnel) to a third host.

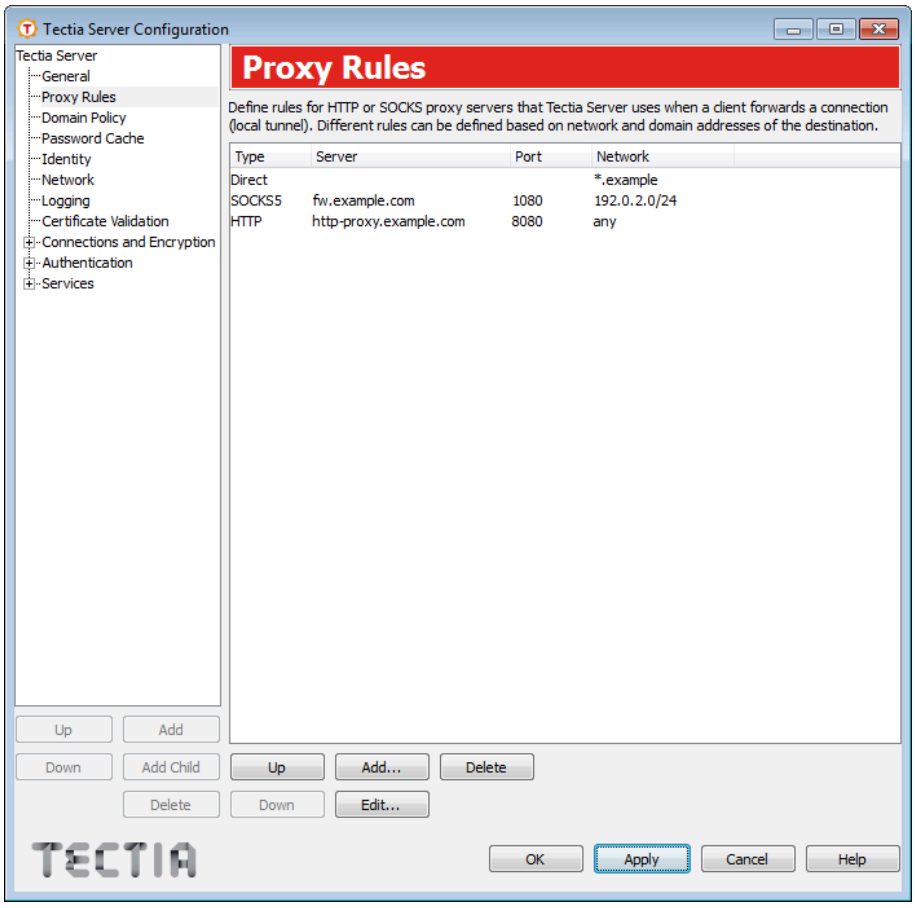


Figure 4.5. Tectia Server Configuration - Proxy Rules page

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**.

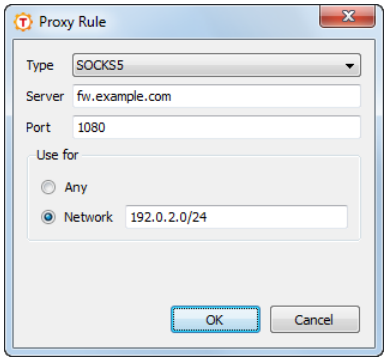


Figure 4.6. Defining a proxy rule

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 host name 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 **Up** and **Down** buttons to change the order of the rules.

4.1.4 Domain Policy

On the **Domain Policy** page you can define how Tectia Server handles the user name when a client user tries to log in without specifying the prefix (indicating a local or domain user account). This setting defines where the server will look for the user account, and how it will fill in the missing prefix part.

On this page you can also define domain user accounts for domain access with one-way trust.

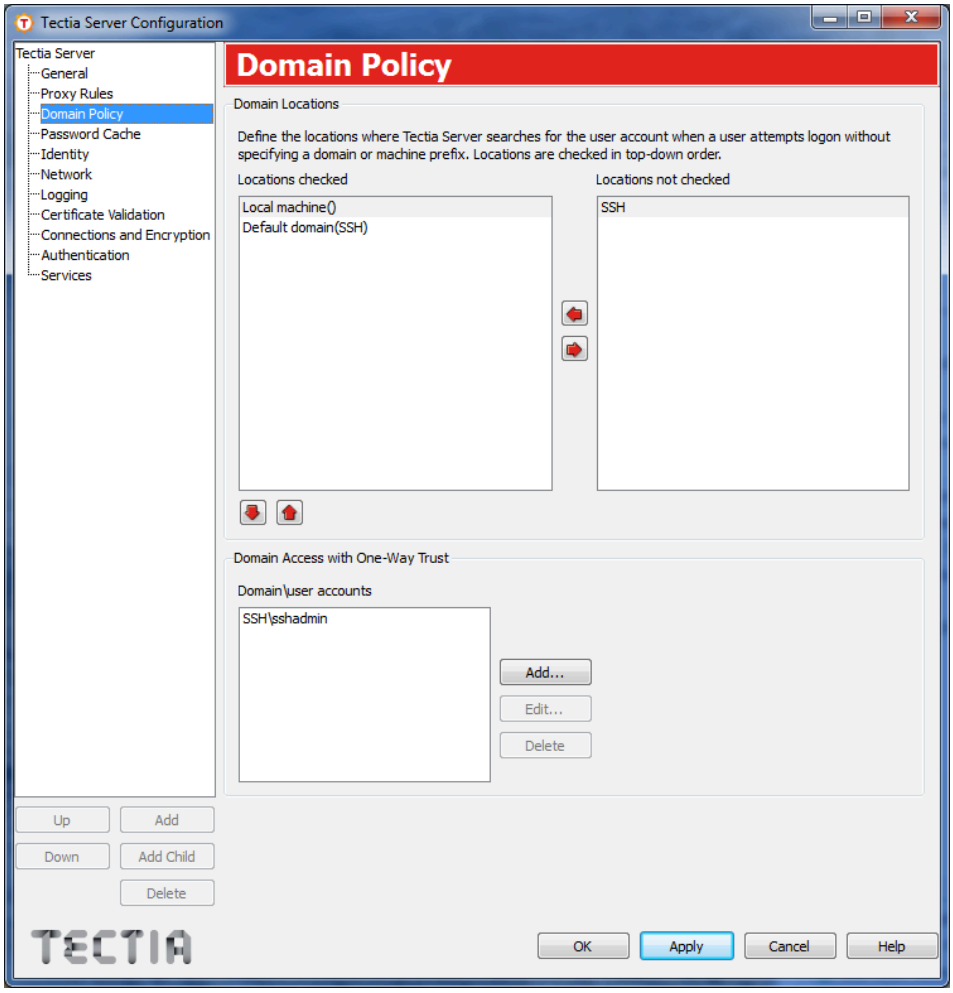


Figure 4.7. Tectia Server Configuration - Domain Policy page

Domain Locations

Tectia Server automatically lists all domains the local machine is part of, and places them in the **Locations not checked** field.

Move the relevant domains to the **Locations checked** field and arrange them to an order of preference. When a user logs in without a prefix, the user name is searched under the listed domains from top down. When a match is found, the rest of the domains are discarded. If no matching user accounts are found, authentication fails.

Option `Default domain` means that a user without a specified prefix will be treated as a domain user, and the default domain name of the local machine is added to the user name (`username` → `defaultdomain_name\username`).

Option `Local machine` means that a user without a specified prefix will be treated as a local user (`username` → `localmachine_name\username`).

You can move unwanted domains to the **Locations not checked** list. These domains are not checked when searching for the user account.

If nothing is defined in the **Locations checked** list, Tectia Server first checks if the user name is valid in the default domain, and if no match is found, the user will be treated as a local user with the local machine name as the prefix.

Domain Access with One-Way Trust

In Windows domains, you can configure Tectia Server for domain access with one-way trust. A one-way trust is a single, non-transitive trust relationship between two domains. In a one-way trust configuration between Tectia Server and a domain controller, the domain controller does not trust the Tectia Server process. The domain controller therefore refuses to give Tectia Server any information about the user that is trying to log on. Because Tectia Server does not know enough about the user, it refuses the logon procedure. You can use a domain user account to get this information from the domain controller.

Note that you can only define one domain user account per domain.

To add a new domain user account for domain access with one-way trust:

1. Click **Add**. The **Domain user information** dialog box opens.
2. Enter the **Domain**, **Username** and **Password** for the account. The password will be stored in the password cache (see Section 4.1.5). Click **OK**.

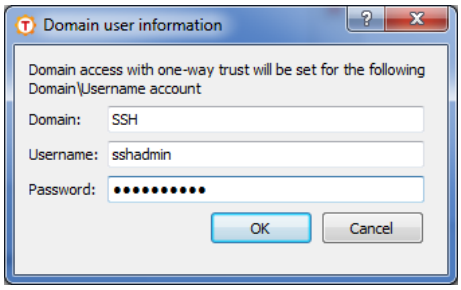


Figure 4.8. Adding a new domain\user account.

To edit an account, select the account from the **Domain\user accounts** list and click **Edit**.

To remove an account, select the account from the **Domain\user accounts** list and click **Delete**.

4.1.5 Password Cache

The **Password Cache** feature is for users who use public-key authentication to log on to Tectia Server on Windows and want to access network resources, for example, shared folders.

When enabled, the password cache stores users' passwords every time they log on to Tectia Server on Windows using password or keyboard-interactive password authentication.

When a user whose password is stored in the cache, logs on using public-key authentication, the password is taken from the cache and used for the logon. The password authentication is performed after the public-key authentication has been successfully completed. From operating system point of view, the user has been logged on using password, and this allows the user to access network resources.

The passwords are stored in encrypted format.

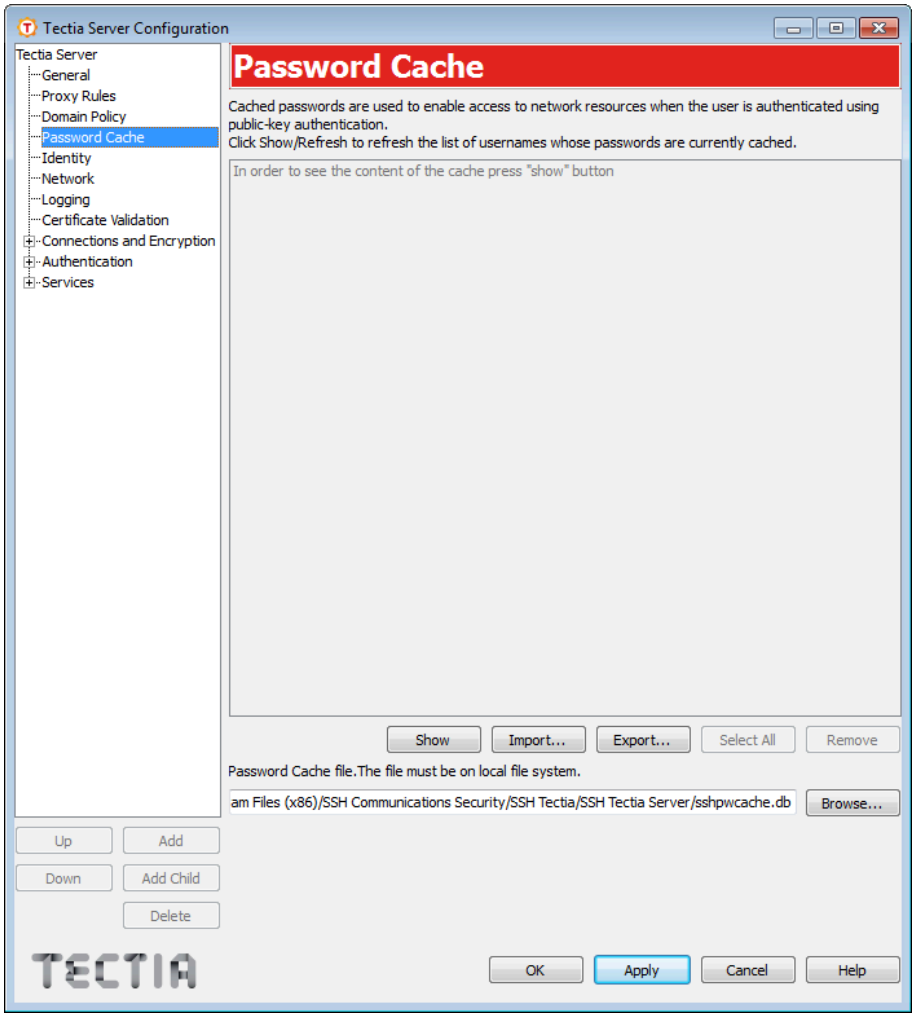


Figure 4.9. Tectia Server Configuration - Password Cache page

To view a list of user names whose passwords are stored in the cache, on the **Password Cache** page, click **Show**. To update the list, click **Refresh**.

To export the current password cache into an external encrypted file:

1. Click **Export**. The **Export Password Database** dialog box opens.

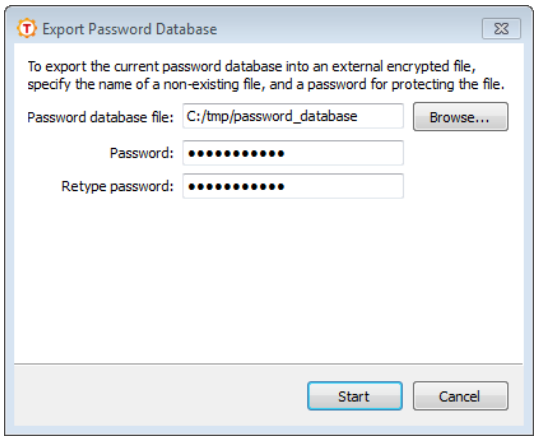


Figure 4.10. Exporting a password database

2. Enter the path to the **Password database file** you want to export the password cache to. The file must reside on a local drive. Existing files will not be overwritten, so if you enter the name of an existing file, the export will fail.

3. Enter the **Password** that will be used to protect the exported password database file. Tectia enforces the use of strong passwords for the password cache export and import functions. Instead of explicit password requirements, we use a "password class" system. For example, a password that consists of eight unique characters from three different character classes or a password of eleven unique characters from two character classes are deemed strong enough. The character classes are: digits, lower-case letters, upper-case letters, and other characters. When calculating the number of different character classes, upper-case letters used as the first character and digits used as the last character of a password are ignored.

4. **Retype password**: Type the password again to ensure you have not made a typing error.

5. Click **Start**. The export operation starts. You will see a notification once the operation has completed.

To import a previously exported password database from an external encrypted file:

1. Click **Import**. The **Import Password Database** dialog box opens.

Figure 4.11. Importing a password database

2. Enter the path to the **Password database file** you want to import. The file must reside on a local drive.

Caution

The passwords of user names that already exist in the current password cache will be overwritten by those in the imported password database file.

3. Enter the **Password** that protects the password database file you want to import.

4. Click **Start**. The import operation starts.

To remove passwords from the cache, select the user name(s) from the list and click **Remove**. The removal cannot be undone (but the password can be cached again by logging on using password authentication).

Password cache file

The password cache must be on local file system since the Tectia Server process must have access to it. The default cache file location is `<INSTALLDIR>\SSH Tectia Server\sshpwcache.db`. You can freely choose any other file location and name.

You can enable or disable the password cache for each authentication rule separately. By default, the password cache is disabled. For more information, see the section called “Parameters”.

4.1.6 Identity

The **Identity** page is used to specify the host keys and host certificates that identify the server to the clients.

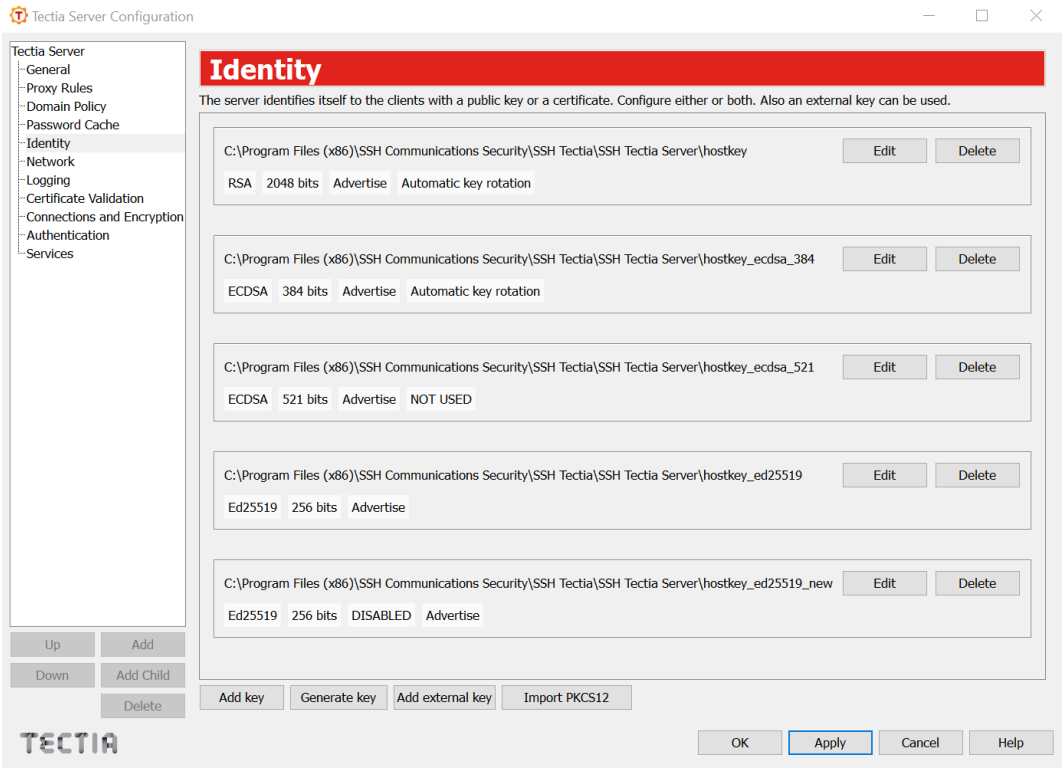


Figure 4.12. Tectia Server Configuration - Identity page

Configured keys are listed here with tags to show some of their features, along with controls to edit or delete them.

Edit

This opens the same host key dialog screen as the Add key button. For more information about the dialog, see the **Add key** section below.

Delete

Remove the selected host-key files from configuration.

Add key

Opens a dialog in which you can add a host key. The same dialog screen opens when you click on the **Edit** key next to a listed key.

You can add a private and/or public host-key file by clicking the **Browse** button next to the associated text field. The **Select File** dialog appears, allowing you to find and specify the desired file. You can also type the path and file name directly into the text field.

The default private-key file is `hostkey`, located in the installation directory ("`<INSTALLDIR>\SSH Tectia Server`", see Section 1.1.2). The private-key file and directory should have full permissions for the **Administrators** group and the **SYSTEM** account and no other permissions.

If the public key is not specified, it will be derived from the private key. However, specifying the public key will decrease the start-up time for the software, as deriving the public key is a fairly slow operation. If the public key is a certificate, the dialog will display a View certificate button.

The dialog will display the key fingerprints in SHA-256, Babble, and RFC 4716 formats.

Under the attributes you can set options for server host-key rotation. Filling in the automatic key-rotation period will enable key rotation for the selected key; once the key-rotation time is reached, the key will be rotated according to standard key-rotation rules. The key-rotation margin will specify for how long the new key will be advertised to the clients before the key is rotated. To learn more about key rotation, see Section 5.2.3.

Generate key

Click the **Generate key** button to generate a new RSA/ECDSA/Ed25519 host key pair. This launches the **ssh-keygen-g3.exe** command-line tool and generates an RSA/ECDSA/Ed25519 key pair. The default length of the generated key pair is 3072 bits for RSA, 384 bits for ECDSA, and 256 bits for Ed25519 keys.

You can generate the key pairs, including deprecated DSA host key if needed, also manually with a command line tool. See instruction in ssh-keygen-g3(1).

Note

Note that the server will only use the first key of a given type as a host key. Different key types can be used as host keys at the same time, but the server only uses the first key of each type as a host key.

Add external key

Opens a dialog in which you can specify an external host key to be used. The fields are **Provider Type** and **Init string**. You can also use Test Scan to attempt adding a `software` or a `pkcs11` provider and scanning it for keys.

For more information on the different external keys and their initialization strings, see **externalkey** in ssh-server-config(5).

Import PKCS12

Click the **Import PKCS12** button to import a private key stored in the Personal Information Exchange (PFX) format. The **Select File** dialog appears, allowing you to specify the desired file.

Note

Notice that all key and certificate files should be located on a local drive. Network or mapped drives should not be used, as the server program may not have proper access rights for them.

See also Section 5.2, Section 5.3, and Section 5.4.

4.1.7 Network

The **Network** page allows you to specify the network interfaces the server is listening for connections.

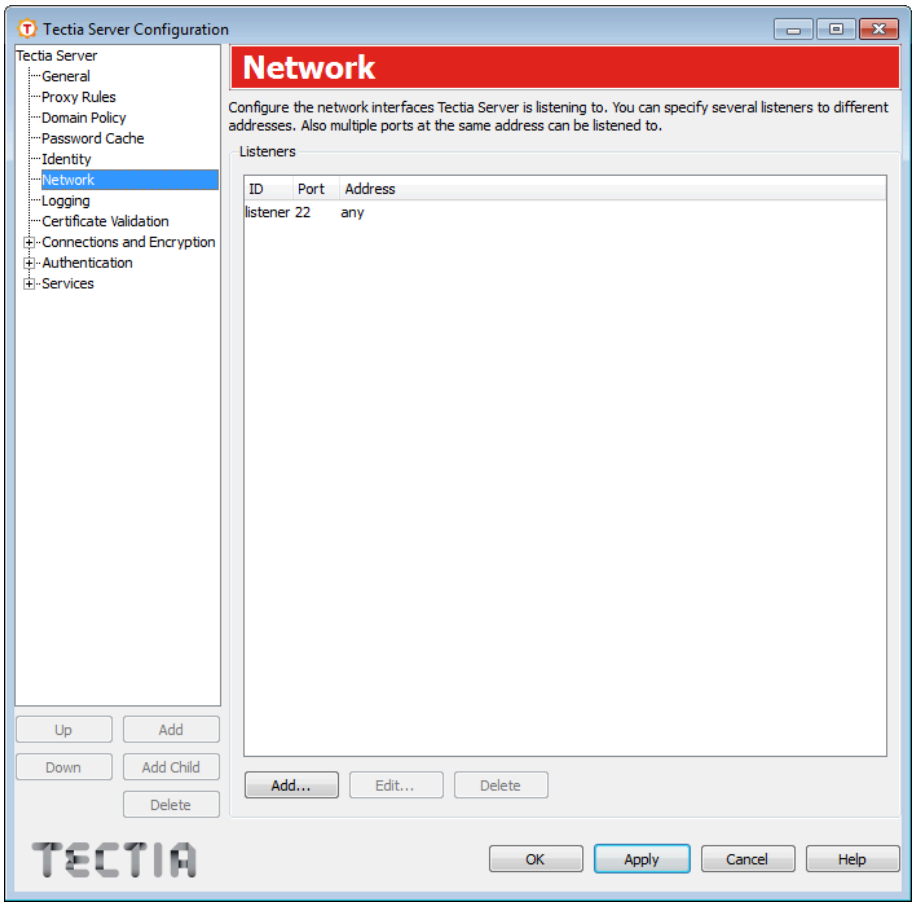


Figure 4.13. Tectia Server Configuration - Network page

The list shows the interfaces Tectia Server is listening on. You can specify several listeners to different addresses. Also multiple ports at the same address can be listened to.

To add a new network listener:

1. Click **Add**. The **Listener** dialog box opens.

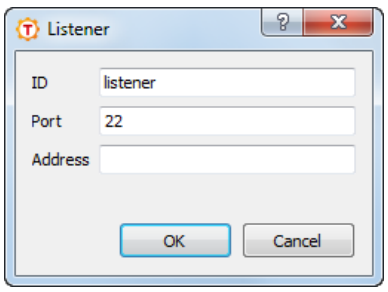


Figure 4.14. Editing a listener

2. Enter the **ID** of the listener. The ID must be unique.

Enter also the **Port** number that the server listens on (allowed values are 1 - 65535). The default port is 22.

Optionally you can also specify the IP **Address** of the network interface card where the Secure Shell server socket is bound. If the address is not specified, the server will listen to the given port on all interfaces.

Click **OK** when finished.

To edit a listener, select the listener from the list and click **Edit**.

To remove a listener, select the listener from the list and click **Delete**.

Note

The server has to be restarted to use the new listener settings.

4.1.8 Logging

The **Logging** page allows you to customize the information that is logged in the event log.

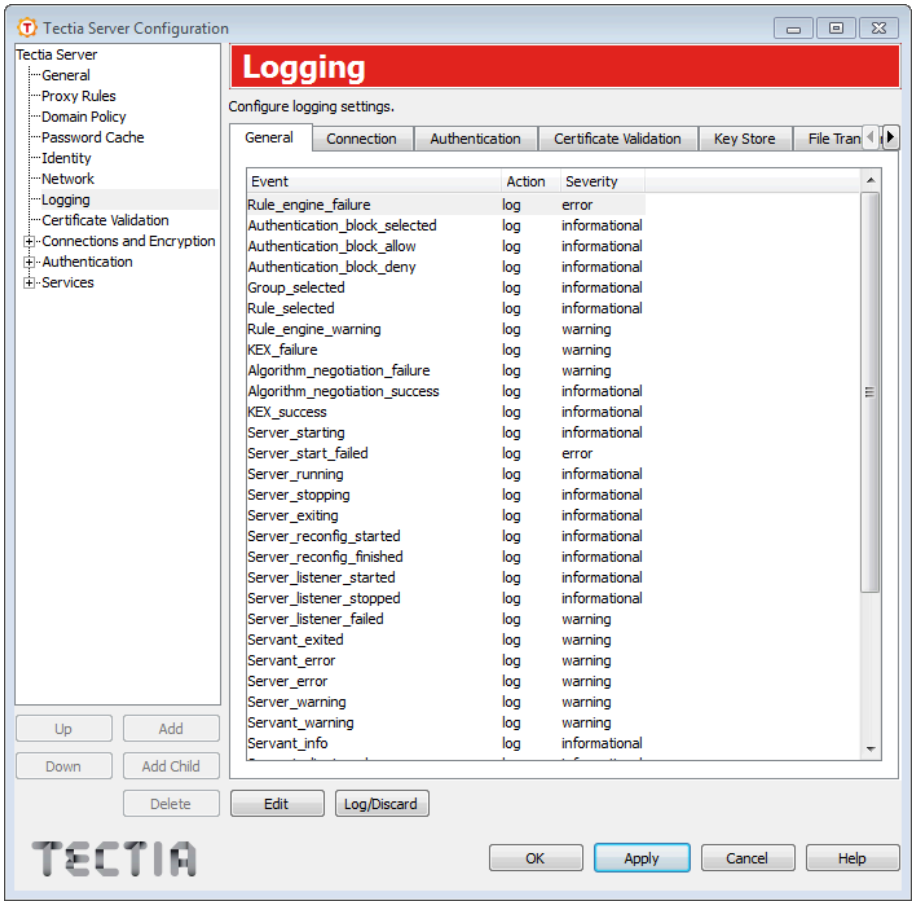


Figure 4.15. Tectia Server Configuration - Logging page

On this page, you can see a list of log events generated by Tectia Server. The events are grouped under tabs according to the event categories. Each event has an associated **Action** and **Severity**. They have reasonable default values, which are used if no explicit logging settings are made.

The action can be either **Log** or **Discard**. Setting the action to **Discard** causes the server to ignore the log event.

Note

If the server fails to start before the server configuration has been read, or because of an error in reading the configuration (for example because of incorrect access permissions or invalid content of the configuration file), the event `Server_start_failed` will be logged even if its **Action** has been set to `Discard`.

On Windows, the following event severities are used:

- **Informational**
- **Warning**
- **Error**
- **Security success**
- **Security failure**

For more information on the event types, see Section 6.2.

For a description of the log events, see Appendix D.

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 severity:

1. Select a log event from the list and click **Edit**. You can select multiple events by holding down the SHIFT or CTRL key while clicking. The **Edit Logging Event** dialog box opens.

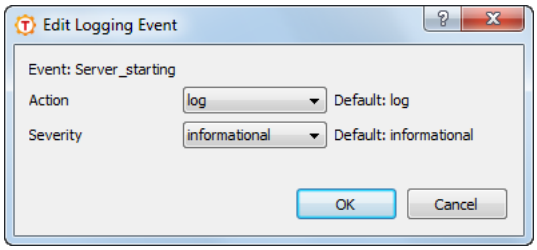


Figure 4.16. Editing a log event

2. Select the **Action** and **Severity** for the event and click **OK**.

4.1.9 Certificate Validation

On the **Certificate Validation** page, you can configure certification authorities (CA) that are trusted in user authentication.

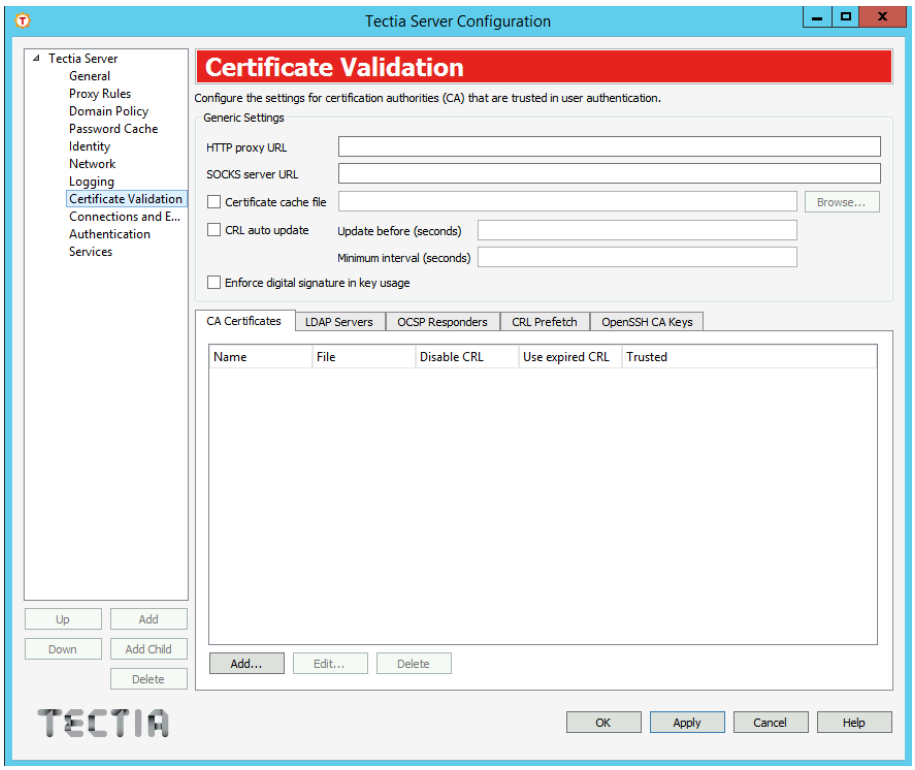


Figure 4.17. Tectia Server Configuration - Certificate Validation page

Generic Settings

Generic settings apply to all CA certificates and CRL fetching.

HTTP proxy URL

Define a HTTP proxy URL if one is required for making LDAP or OCSP queries for certificate validity.

The format of the URL is as follows:

```
http://username@proxy_server:port/network/netmask,network/netmask ...
```

The HTTP proxy address is given first and after it the networks that are connected directly (without the proxy).

SOCKS server URL

Define a SOCKS server URL if one is required for making LDAP or OCSP queries for certificate validity.

The format of the URL is as follows:

```
socks://username@socks_server:port/network/netmask,network/netmask ...
```

The SOCKS server address is given first and after it the networks that are connected directly (without the SOCKS server).

Certificate cache file

Select the check box to enable certificate caching.

Click the **Browse** button to select the cache file where the certificates and CRLs are stored when the Tectia Server service is stopped, and read back in when the service is restarted. The **Select File** dialog appears, allowing you to specify the desired file. You can also type the path and file name directly into the text field.

CRL auto update

Select the check box to enable automatic updating of certificate revocation lists.

When auto update is on, Tectia Server periodically tries to download the new CRL before the old one has expired. The **Update before** field specifies how many seconds before the expiration the update takes place. The **Minimum interval** field sets a limit for the maximum update frequency. The default minimum interval is `30` seconds.

Enforce digital signature in key usage

One of the compliance requirements of the US Department of Defense Public-Key Infrastructure (DoD PKI) is to have the Digital Signature bit set in the Key Usage of the certificate. To fulfill the compliance requirement by enforcing digital signature in key usage, select this check box.

LDAP Servers

On the **LDAP Servers** tab, you can define LDAP servers that are used for fetching certificate revocation lists (CRLs) and/or subordinate CA certificates based on the issuer name of the certificate being validated.

If a CRL distribution point is defined in the certificate, the CRL is automatically retrieved from that address.

To add an LDAP server, click **Add**. The **LDAP Server** dialog box opens. Enter the **Address** and **Port** of the server and click **OK**. The default port is `389`.

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**.

OCSP Responders

On the **OCSP Responders** tab, you can define OCSP responder servers that are used for Online Certificate Status Protocol queries.

For the OCSP validation to succeed, both the end-entity certificate and the OCSP responder certificate must be issued by the same CA unless the optional Responder certificate is configured to enable trusted mode OCSP. For more information, see **ocsp-responder** .

If the certificate has an `Authority Info Access` extension with an OCSP Responder URL, it is only used if there are no configured OCSP responders. It is not used if any OCSP responders have been configured.

To add an OCSP responder, click **Add**. The **OCSP Responder** dialog box opens. Enter the HTTP **URL** of the server. Optionally, you can also configure a **Responder certificate** to enable trusted mode OSCP or a **Validity period** in seconds for the OCSP data. During this time, new OCSP queries for the same certificate are not made but the old result is used. Click **OK** when finished.

If an OCSP 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.

To edit an OCSP responder, select the responder from the list and click **Edit**.

To delete an OCSP responder, select the responder from the list and click **Delete**.

CRL Prefetch

On the **CRL Prefetch** tab, you can define addresses from which CRLs are periodically downloaded.

To add a CRL prefetch address, click **Add**. The **CRL Prefetch** dialog box opens. Enter the **Interval** how often the CRL is downloaded and the **URL** of the CRL distribution point and click **OK**. The default download interval is `3600` (seconds).

The URL 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. Enter the file URL in this format:

```
file:///absolute/path/name
```

To edit a CRL prefetch address, select the address from the list and click **Edit**.

To delete a CRL prefetch address, select the address from the list and click **Delete**.

CA Certificates

On the **CA Certificates** tab, you can define the CA certificates that are trusted for user authentication, as well as intermediate CA certificates.

To add a CA certificate:

1. Click **Add**. The **CA Certificate** dialog box opens.

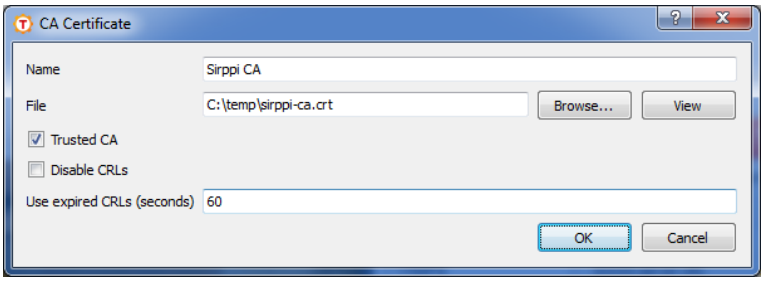


Figure 4.18. Editing CA certificate settings

2. Enter the **Name** of the CA. The CA **Name** can be referred to in the selectors on the Authentication page. See Section 4.1.12.

3. Click the **Browse** button on the right-hand side of the text field to locate a CA certificate file. The **Select File** dialog appears, allowing you to specify the desired file. You can also type the path and file name directly in the text field.

   Click the **View** button to display the currently selected CA certificate.

4.

5.

To edit a CA, select the CA from the **CA Certificates** list and click **Edit**.

To remove a CA from the **CA Certificates** list, select the CA and click **Delete**.

OpenSSH CA Keys

On the **OpenSSH CA Keys** tab, you can define the OpenSSH CA keys that are trusted for user authentication.

To add an OpenSSH CA key, click the **Add** button, and provide a name for the list entry, and the key file.

To edit an OpenSSH CA key entry, select an entry from the list, and click the **Edit** button.

To delete an OpenSSH CA key entry, select an entry from the list, and click the **Delete** button.

4.1.10 Defining Access Rules Using Selectors (Advanced Mode)

When the **Tectia Server Configuration** tool is run in the advanced GUI mode, the **Connections and Encryption**, **Authentication**, and **Services** pages can contain several sub-pages, each of which defines its own set of access rules. The rule to be used in each case is chosen using *selectors*.

Selectors define the access rules for users based on the user parameters such as user name or location. Users can be divided to groups dynamically, for example, based on the authentication method they used for logging in. On the **Services** page, each group can then be allowed or denied services such as tunneling, file transfer, or terminal access.

Use the **Add** and **Delete** buttons below the tree view to add and delete rules. Each rule will have a sub-page with two or more tabs. On the **Selectors** tab, you can edit the selectors of the rule, and on the other tab(s), you can configure the settings for the rule.

Under **Authentication**, you can also add child authentication methods using the **Add Child** button.

Whenever a user is attempting login to the server, the connections, authentication, and services rules are processed in top-down order. In each case, the first rule that matches the user is used. Use the **Up** and **Down** buttons to change the order of the rules. See Section 4.2.2 for more information on selector processing.

The commands for adding, deleting, and moving rules are also available from a shortcut menu (right-click on a rule in the tree view).

Editing Selectors

The selectors can be edited on the **Selectors** tab of the **Connections and Encryption**, **Authentication**, and **Services** sub-pages.

The **Selectors** tab shows a list of all selectors and attributes that apply to the rule (connection, authentication, or service group rule, depending on the page you are on).

The selector elements are numbered. If any of the selectors match, the rule will match and is used.

Each selector element can have one or more attributes. All attributes of the selector must match for the selector to match, except with the attributes of the same type, of which only one has to match.

To add a new selector to the rule, click **Add Selector**. The new selector will contain automatically at least one attribute. To add a new attribute to a selector, choose a selector from the list and click **Add Attribute**. In both cases, the **Add Selector** dialog box opens allowing you to specify the selector type. See Figure 4.19.

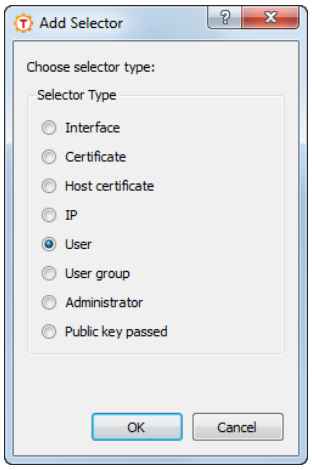


Figure 4.19. The Add Selector dialog box

Select the selector type and click **OK**.

The attributes of the selector depend on the type. The different selector types are described below.

Interface

The Interface selector is matched to the listener interface **ID** or **Address** and/or **Port**. At least one attribute must be given. If the **ID** is defined, the others MUST NOT be given. If the **ID** is not defined, either or both of **Address** and **Port** may be given.

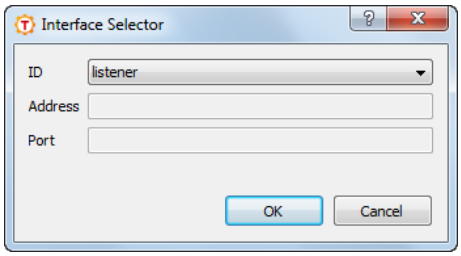


Figure 4.20. The Interface Selector dialog box

Certificate

This selector matches a **Pattern** in a specified **Field** of the user certificate. Using this selector requires that the parent rule in the authentication chain enables public-key authentication.

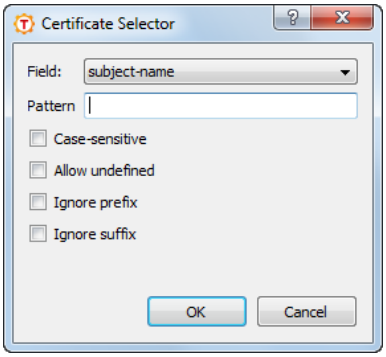


Figure 4.21. The Certificate Selector dialog box

The field can be either **ca-list**, **issuer-name**, **subject-name**, **serial-number**, **altname-email**, **altname-upn**, **altname-ip**, or **altname-fqdn**.

The format of the pattern depends on the type of the field. The **ca-list** field contains a list of CA names separated by commas. The names that are defined in the `ca-certificate` element in `ssh-server-config.xml` are used. The **issuer-name** and **subject-name** fields contain distinguished names, **serial-number** a positive integer. The **altname-fqdn** field contains a host name and **altname-ip** an IP address or a range. The **altname-email** field contains an email address and **altname-upn** the principal name.

The **altname-fqdn**, **altname-upn**, **altname-email**, **subject-name**, and **issuer-name** selectors may contain the `%username%` keyword which is replaced with the user's login name before comparing with the actual certificate data. For domain accounts, the `%username-without-domain%` keyword can be used and it is replaced by the user's login name without the domain part. The `%hostname%` keyword can be used in the same way and it is replaced by the client's FQDN. These patterns may also contain "*" and "?" globbing characters.

Patterns are normally matched case-insensitively. Select the **Case-sensitive** check box to match the pattern case-sensitively.

For the **issuer-name** and **subject-name** selectors, you can also define if the pattern has to match the subject name completely or only partly. Select the **ignore-prefix** check box to match only the end of the subject name. Select the **ignore-suffix** check box to match only the beginning of the subject name. By default, the ignore options are unselected.

You can also select both of the ignore options simultaneously in which case the pattern has to match with some point in the subject name. For example: when both ignore settings are selected, pattern `O=SSH,OU=*,CN=example` matches with:

```
C=FI, O=SSH, OU=RandD, CN=example, CN=UID12345
```

Normally if the certificate field to be matched is not available, the selector matching process ends in error. However, if the **Allow undefined** check box is selected, the undefined field is treated as non-matched and the matching continues to other selectors. For more information, see the section called "Selectors and Undefined Data".

Caution

When creating the certificate selectors, make sure that every selector element ties the user name to the certificate, either by including a **User** selector attribute, or by putting the special substitution string `%username%` or `%username-without-domain%` to a field used to match the corresponding field in the certificate.

Failing to do this may cause unintended consequences, for example authentication succeeding with many different user names with a single certificate.

Host certificate

This selector matches a **Pattern** in a specified **Field** of the client host certificate. Using this selector requires that the parent rule in the authentication chain enables host-based authentication.

The field can be either **ca-list**, **issuer-name**, **subject-name**, **serial-number**, **altname-email**, **altname-upn**, **altname-ip**, or **altname-fqdn**.

Patterns are normally matched case-insensitively. Select the **Case-sensitive** check box to match the pattern case-sensitively.

For the **subject-name** selector, you can also define if the pattern has to match the subject name completely or only partly. Select the **ignore-prefix** check box to match only the end of the subject name. Select the **ignore-suffix** check box to match only the beginning of the subject name. You can also select both of the ignore options simultaneously in which case the pattern has to match with some point in the subject name. By default, the ignore options are unselected.

Normally if the certificate field to be matched is not available, the selector matching process ends in error. However, if the **Allow undefined** check box is selected, the undefined field is treated as non-matched and the matching continues to other selectors. See the section called "Selectors and Undefined Data" for more information.

IP

The IP selector matches an IP Address or fully qualified domain name (FQDN) of the client.

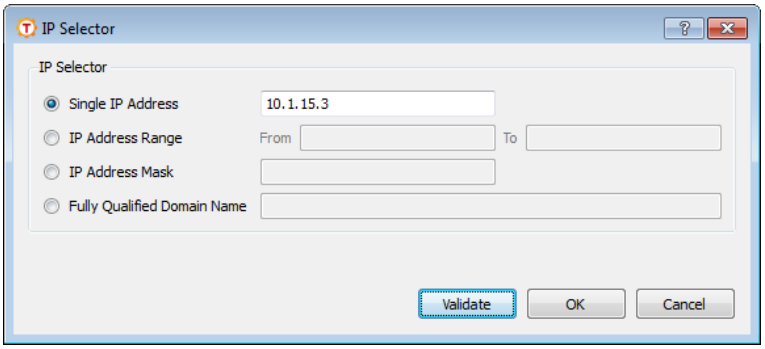


Figure 4.22. The IP Selector dialog box

The IP address can be in one of the following formats:

- 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 fully qualified domain name is matched to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns. These patterns may also contain "*" and "?" globbing characters. The form of the pattern is not checked.

After entering the IP address, you can click the **Validate** button to check whether the format of the address is valid. The validate feature is available only for single IP addresses and IP address ranges.

User

This selector matches a user **Name**. A list of user names can be given as a comma-separated list.

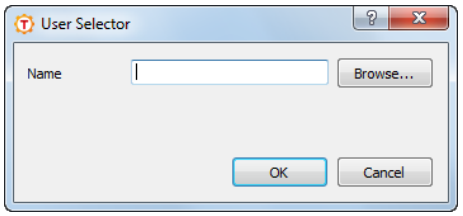


Figure 4.23. The User Selector dialog box

Names are matched case-insensitively.

Note

We recommend using the object picker dialog in the GUI when defining the selectors, because it returns the correct form of user names and host names. To open the object picker, click the **Browse** button in the User Selector dialog.

If the original user name is longer than 20 characters, Windows stores the name in both full format and in short format with max 20 characters. Similarly, long host names are cut to 15 characters.

When Tectia Server is running in domain environment on Windows, the user names and host names must be used in the short format in the selectors. For example, user name `longusername1234567890123` (25 chars) cannot be used as such in the Tectia Server selectors, instead the user name is used in the short form as follows:

```
domain\longusername12345678
```

Note that Tectia Server supports only the following user name format in selectors:

```
domain\username
```

The UPN format `username@domain.com` is *not* supported.

To browse for Windows domain user names directly from an Active Directory server, follow these instructions:

1. Click **Browse**. This opens a standard Windows **Select Users** dialog box that allows you to search for user names from a directory server.

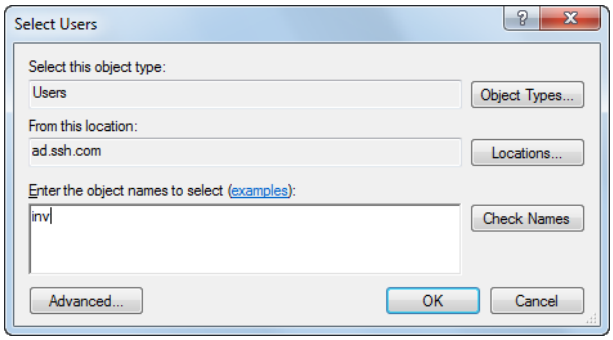


Figure 4.24. Selecting users from Active Directory

2. Click **Locations** to select the Active Directory server you want to use. Select the server from the list and click **OK**.

3. Enter the user name or a part of it in the text field. You can enter several names and separate them with semicolons. Click **Check Names** to check the names from the Active Directory server.

   To use advanced search options, click **Advanced**. This opens an advanced search dialog.

4. After you have found the user name(s), click **OK** to return to the **User Selector** dialog box. The selected domain user accounts are now shown in the **Name** field.

User group

This selector matches a user group **Name**. A list of user-group names can be given as a comma-separated list.

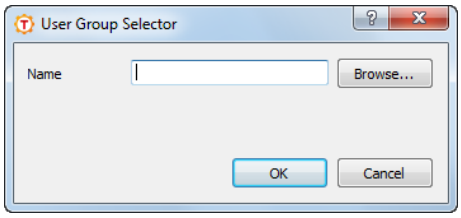


Figure 4.25. The User Group Selector dialog box

Names are matched case-insensitively.

On Windows domain environment, the `user` and `user-group` selectors have a length limitation. For more information, see the description of option **User** above.

To browse for Windows domain user groups directly from an Active Directory server, follow these instructions:

1. Click **Browse**. This opens a standard Windows **Select Groups** dialog box that allows you to search for user group names from a directory server.

2. Click **Locations** to select the Active Directory server you want to use. Select the server from the list and click **OK**.

3. Enter the group name or a part of it in the text field. You can enter several names and separate them with semicolons. Click **Check Names** to check the names from the Active Directory server.

   To use advanced search options, click **Advanced**. This opens an advanced search dialog.

4. After you have found the user group name(s), click **OK** to return to the **User Group Selector** dialog box. The selected domain user groups are now shown in the **Name** field.

Administrator

This selector matches a privileged user (administrator) or a non-privileged user.

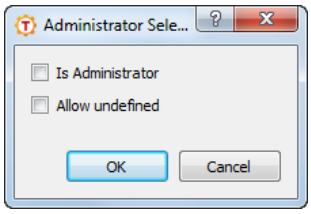


Figure 4.26. The Administrator Selector dialog box

Select the **Is Administrator** check box to match the selector to a privileged user or clear the checkbox to match it to a normal user.

If this selector is used in an authentication rule and the user is logging in using a domain account and does not yet have an access token allocated, the selector matching process ends in error. However, if

the **Allow undefined** check box is selected, the selector is treated as non-matched and the matching continues to other selectors. For more information, see the section called “Selectors and Undefined Data”.

Note

The user-privilege level is not available during the authentication phase when the user is logging in using a domain account and does not yet have an access token allocated. To get the user-privilege status for domain users, the user should first pass password or GSSAPI authentication.

If the privilege level needs to be checked for local accounts, the **Allow undefined** check box should be selected or else connection fails for users logging in using domain accounts. However, this means that the user-privilege status will not be verified for Windows domain users.

To check the privilege level of domain accounts on a Windows server in the authentication phase, the **Administrator** selector should be used in a nested authentication rule when password or GSSAPI authentication has already been passed.

Public key passed

This selector matches if authentication is passed using a normal public key (without a certificate).

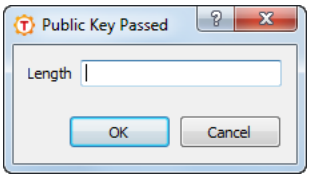


Figure 4.27. The Public Key Passed Selector dialog box

Optionally, the **Length** range of the public key can be given, for example `2048-4096`.

4.1.11 Connections and Encryption

On the **Connections and Encryption** page, you can create connection rules that restrict connections based on various selectors. You can also set the ciphers, MACs and KEXs used for the connections.

The selectors define which connections a connection rule applies to. The order of the rules is important. The first matching rule is used and the remaining rules are ignored.

If no selectors (or only empty selectors) are specified in a connection rule, the rule matches all connections. In the simple GUI mode, there is only one connection rule that is used for all connections.

If a user does not match any selectors in the connection rules, the connection is allowed with server default connection settings.

To add a new connection rule, click the **Add** button below the tree view. Each rule will have a sub-page with two tabs. On the **Selectors** tab, you can edit the selectors of the rule and define whether the connection is allowed or denied, and on the **Parameters** tab, you can configure the settings for the rule.

To edit a connection rule, select a connection item on the tree view. For more information, see the section called “Editing Connection Rules”.

To change the order of the rules, select a connection item on the tree view and use the **Up** and **Down** buttons. The rules are read in order, and the first matching connection rule on the list is used.

To delete a connection rule, select a connection item and click **Delete**.

Editing Connection Rules

Each item under **Connections and Encryption** has two tabs, **Selectors** and **Parameters**. The **Selectors** tab is shown only in the advanced GUI mode.

Selectors (Advanced Mode)

On the **Selectors** tab, you can configure the selectors that apply to the connection rule and define whether the connection is allowed or denied.

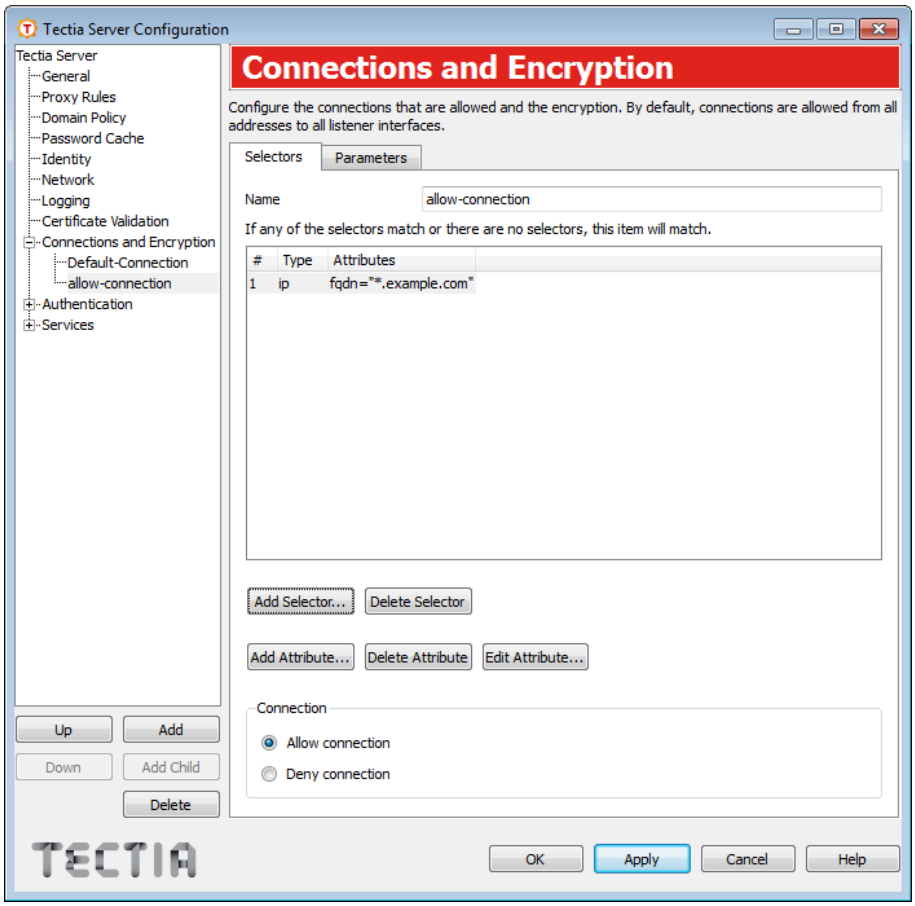


Figure 4.28. Tectia Server Configuration - Connections and Encryption page - Selectors tab

Name

Enter a name for the connection rule.

Selector list view

The selector list view shows the selectors that apply to the rule.

To add a new selector to the rule, click **Add Selector**. The new selector will contain automatically at least one attribute. The **Add Selector** dialog box opens allowing you to specify the selector type. For more information on the different selector attributes, see the section called “Editing Selectors”.

Only the **Interface** and **IP** selector attributes are relevant for connection rules. For example, the user name is not yet available when the connection rules are processed. For more information, see Section 4.2.2.

To remove a selector, choose the selector from the list view on the **Selectors** tab and click **Delete Selector**. This will delete the selector and all its attributes.

To add a new attribute to a selector, choose a selector from the list and click **Add Attribute**. The **Add Selector** dialog box opens. For more information on the different selector attributes, see the section called “Editing Selectors”.

To edit a selector attribute, choose the attribute from the list and click **Edit Attribute**. The relevant selector dialog box opens. For more information on the different selector attributes, see the section called “Editing Selectors” .

To remove a selector attribute, choose the attribute from the list and click **Delete Attribute**. Note that a selector with no attributes will match everything.

Connections

Select whether the connection is allowed or denied.

If you select to deny the connection, the **Parameters** tab is disabled.

Parameters

On the **Parameters** tab, you can configure the allowed ciphers, MACs, host key algorithms and KEXs for the connection.

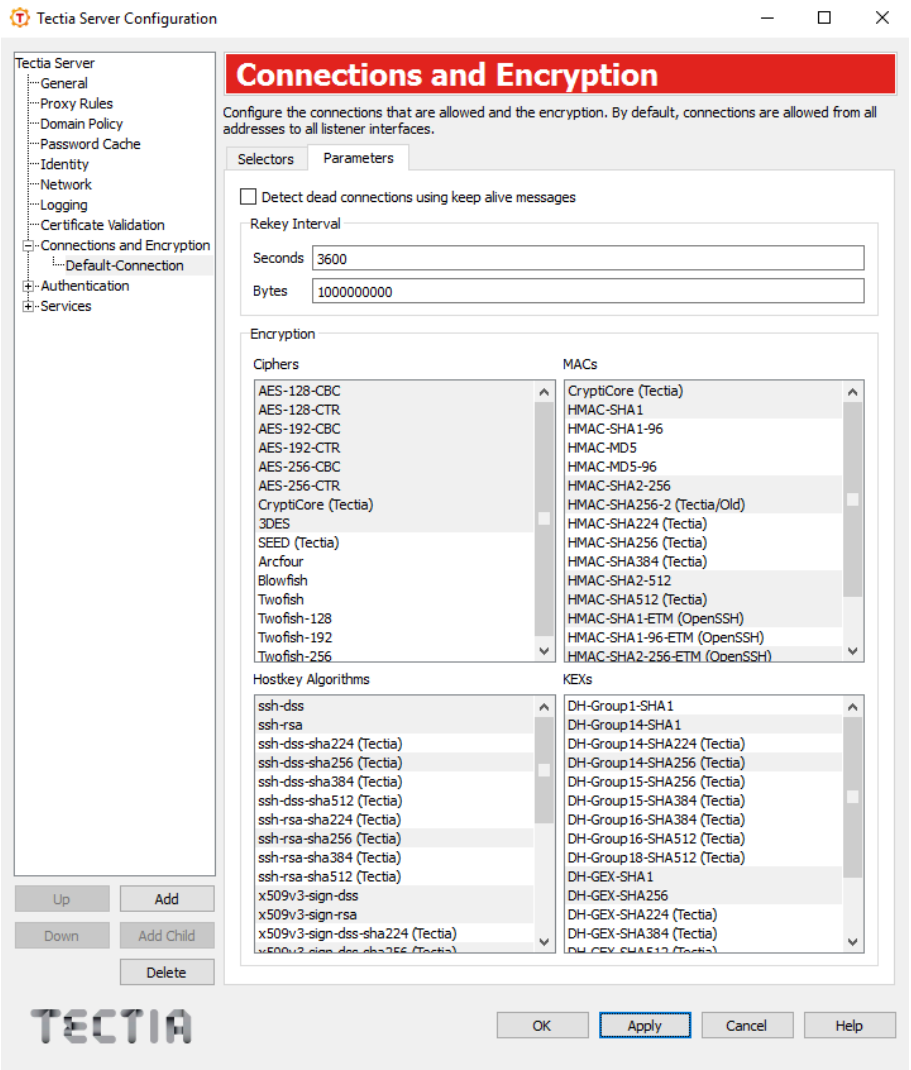


Figure 4.29. Tectia Server Configuration - Connections and Encryption page - Parameters tab

Detect dead connections using keep alive messages

Select this check box to send keep alive messages to the other side. If they are sent, a broken connection or crash of one of the machines will be properly noticed. This also means that connections will die if the route is down temporarily.

Rekey Interval

Specify the number of **Seconds** or transferred **Bytes** after which the key exchange is done again.

If a value for both **Seconds** and **Bytes** is specified, rekeying is done whenever one of the values is reached, after which the counters are reset.

The defaults are `3600` seconds (1 hour) and `1000000000` bytes (~1 GB). The value `0` (zero) turns rekey requests off. This does not prevent the client from requesting rekeys.

Encryption

Under **Encryption**, select the **Ciphers**, **MACs**, **Host key algorithms** and **KEXs** allowed for the connection from the list. To deselect an already selected algorithm, click on it again.

The default ciphers, MACs, host key algorithms and KEXs are marked in the list initially with a gray background.

Tectia proprietary algorithms are marked with **(Tectia)** and are operable with Tectia products only. They correspond to the algorithms that end with `@ssh.com` in the server configuration file.

Ciphers

The list of supported ciphers can be found in Section G.1.

MACs

The list of supported MACs can be found in Section G.3

Host key algorithms

The list of supported host key algorithms can be found in Section G.4.

KEXs

The list of supported KEX methods can be found in Section G.2.

4.1.12 Authentication

On the **Authentication** page you can configure the allowed and required user authentication methods.

Authentication options are specified as chains of authentication rules. In the Simple GUI mode, there is only one authentication rule that is used for all connections. In the Advanced GUI mode, the view always contains the **Default-Authentication** rule, but the administrator can define more rules according to need.

An authentication rule can include one or more selectors and different authentication methods. The selectors define to which users an authentication rule applies. If no selectors (or only empty selectors) are specified in an authentication rule, the rule matches all users.

An authentication rule may also include other authentication rules, forming an authentication chain. When authentication rules are nested within each other, the child rules are interpreted as *required* (all must be passed for the authentication to succeed). You can set multiple authentication methods in the same authentication rule, and the methods are interpreted as *optional* (one of the methods must be passed for the authentication to succeed).

The order of the rules is important. Out of the rules on the same level, the first matching rule is used and the remaining rules are ignored. If the rule has nested child rules, they are matched next using the same procedure.

For more information on authentication chains, see Section 5.12.

To add a new authentication rule, click the **Add** button below the tree view. Each rule will have a sub-page with two tabs. On the **Selectors** tab, you can edit the selectors of the rule and define whether the authentication is allowed or denied, and on the **Parameters** tab, you can configure the settings for the rule.

To edit an authentication rule, select an authentication item on the tree view. For more information, see the section called “Editing Authentication Items”.

To change the order of the rules, select an authentication item on the tree view and use the **Up** and **Down** buttons.

To add a child authentication rule, select an authentication item on the tree view and click the **Add Child** button.

To delete an authentication rule, select an authentication item and click **Delete**.

Editing Authentication Items

Each item under **Authentication** has two tabs, **Selectors** and **Parameters**. The **Selectors** tab is shown only in the advanced GUI mode.

Selectors (Advanced Mode)

On the **Selectors** tab, you can configure the selectors that apply to the authentication rule and define whether the result of the rule is allow or deny.

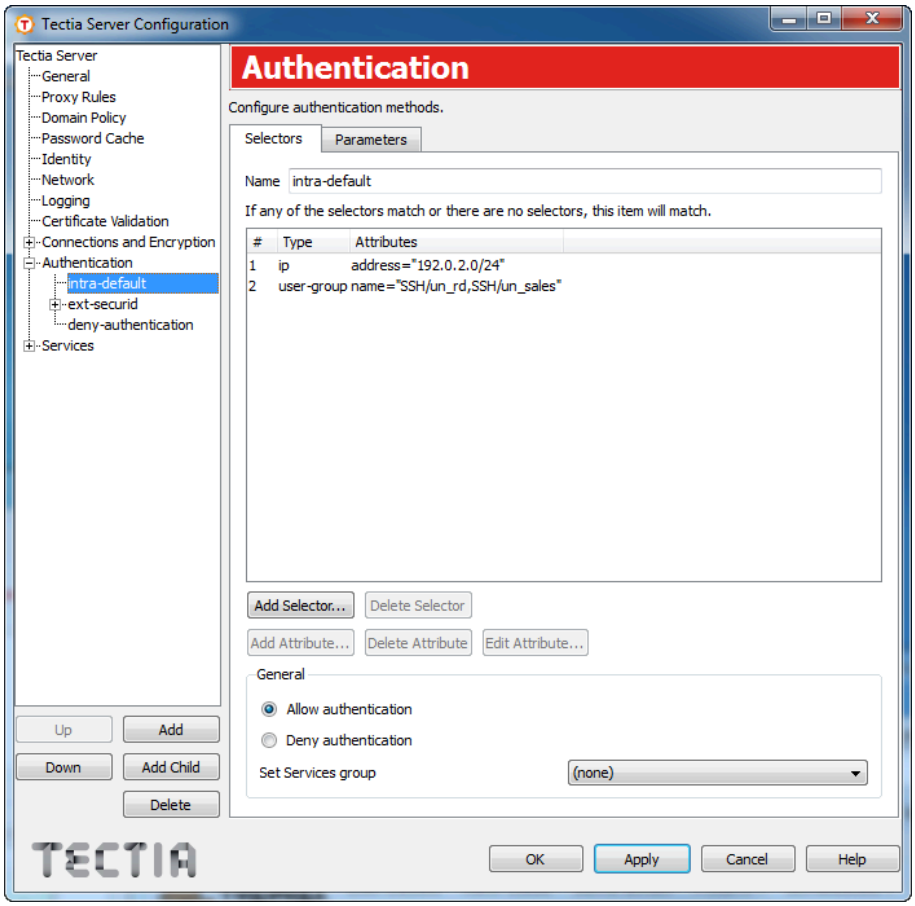


Figure 4.30. Tectia Server Configuration - Authentication page - Selectors tab

Name

Enter a name for the authentication rule.

Selector list view

The selector list view shows the selectors that apply to the rule.

To add a new selector to the rule, click **Add Selector**. The new selector will contain automatically at least one attribute. The **Add Selector** dialog box opens allowing you to specify the selector type. For more information on the different selector attributes, see the section called “Editing Selectors”.

To remove a selector, choose the selector from the list view on the **Selectors** tab and click **Delete Selector**. This will delete the selector and all its attributes.

To add a new attribute to a selector, choose a selector from the list and click **Add Attribute**. The **Add Selector** dialog box opens. For more information on the different selector attributes, see the section called “Editing Selectors”.

To edit a selector attribute, choose the attribute from the list and click **Edit Attribute**. The relevant selector dialog box opens. For more information on the different selector attributes, see the section called “Editing Selectors”.

To remove a selector attribute, choose the attribute from the list and click **Delete Attribute**. Note that a selector with no attributes will match everything.

General

Select whether authentication is allowed or denied.

If an authentication chain ends in a deny action, or if the user does not match any selectors in the authentication rules, the user is not allowed to log in.

In a nested chain of authentication rules, it is possible, for example, to set the parent rule to deny authentication and a child rule with a selector to allow authentication. If the user name matches the selector and successfully completes the authentication method(s), login is allowed.

For more information on the authentication chains, see Section 5.12 .

Set Services group

You can optionally select a group name in the **Set Services group** field. This sets a group for the users that pass the particular authentication chain. The group definition is later used when defining the allowed services for the user.

If the group is set here, it overrides any group selectors on the **Services** page. See Section 4.1.13.

Parameters

On the **Parameters** tab, you can configure which authentication methods are allowed, and how they are used.

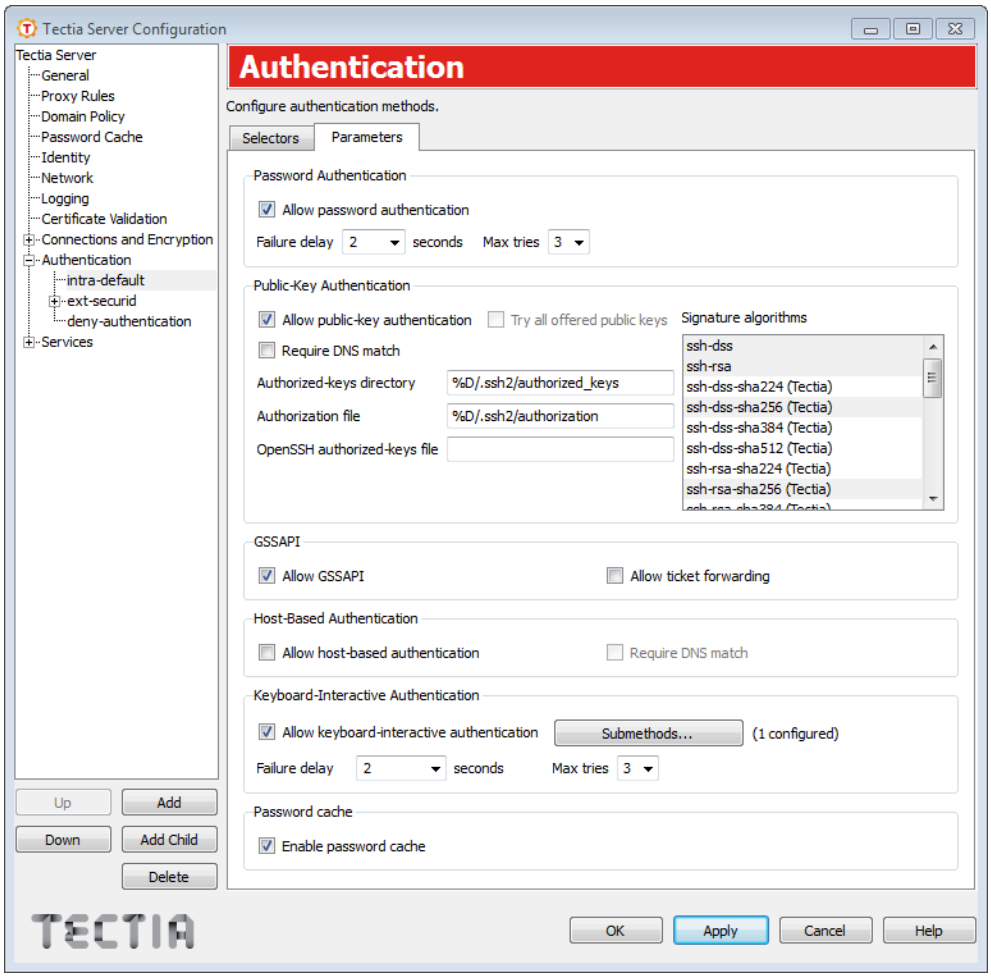


Figure 4.31. Tectia Server Configuration - Authentication page - Parameters tab

Password Authentication

Select the **Allow password authentication** check box if you want to allow password authentication. For more information, see Section 5.5.

Failure delay

Set a delay (in seconds) between a failed attempt and a retry. The default delay is 2 seconds.

Max tries

Set the maximum number of authentication attempts. By default, 3 attempts are allowed.

Public-Key Authentication

Select the **Allow public-key authentication** check box when you want to allow public-key authentication. For more information, see Section 5.6 and Section 5.7.

Try all offered public keys

This option can be used when the authentication rule contains a child rule with certificate selectors.

Select the **Try all offered public keys** check box when you expect the user to have several certificates of which only some allow logon (that is, match the selectors in the child authentication rule).

If the check box is not selected, Tectia Server will try to match only the first certificate offered by the client. If the check box is selected, Tectia Server will try all offered certificates until a match is found.

Require DNS match

Select the check box to require that the host name given by the client matches the one found in DNS. If the host name does not match, the authentication fails. This corresponds to the `require-dns-match` attribute in the server configuration file, see **auth-publickey** .

Authorized-keys directory

Specify a path to the directory that contains the user public keys that are authorized for login. As with the **Authorization file**, the path can contain a pattern string that is expanded by Tectia Server. See the options below. The default is `%D/.ssh2/authorized_keys`.

Authorization file

Specify a path to the file that lists the user public keys that are authorized for login. The path can contain a pattern string that is expanded by Tectia Server.

The following pattern strings can be used:

- `%D` or `%homedir%` is the user's home directory
- `%U` or `%username%` is the user's login name

  For Windows domain users:

  - `%U` is expanded to `domain.username`
  - `%username%` is expanded to `domain\username`

  For local server machine users:

  - `%U` is expanded to `username`
  - `%username%` is expanded to `username` (without the domain prefix)
- `%username-without-domain%` is the user's login name without the domain part.

The default is `%D/.ssh2/authorization`.

For more information on the syntax of the authorization file, see the section called "Authorization File Options".

OpenSSH authorized-keys file

Optionally specify a path to an OpenSSH-style `authorized_keys` file that contains the user public keys that are authorized for login. As above, the path can contain a pattern string that is expanded by Tectia Server.

Signature algorithms

Select the public-key signature algorithms used for user authentication. To deselect an already selected algorithm, click on it again.

The supported and default public-key signature algorithms are the same as those listed for host key algorithms. See **Host key algorithms** .

These authorization file and public-key directory settings will override the **User configuration directory** setting made in the **General** view.

Tectia Server looks for a matching public-key in the following order:

1. In the file defined in **Authorization file**
2. In the directory defined in **Authorized-keys directory**, if no authorization file is available or reading it fails.
3. In the filed defined in **OpenSSH authorized-keys file**, if no matching key was found in the Tectia-related authorization file or key directory.
4. In the **User configuration directory** defined in the **General** view, if none of the above locations produced a matching key.
5. In the default public-key storage location, if no setting was made in **User configuration directory** in the **General** view.

GSSAPI

Select the **Allow GSSAPI** check box to allow GSSAPI authentication. See Section 5.10 for more information.

Allow ticket forwarding

Select the check box to allow forwarding the Kerberos ticket over several connections.

Note

On Microsoft Windows version 5.2 (Server 2003) and newer the possibility to allow Kerberos ticket forwarding is determined by the domain's Kerberos policy. For more information, see "How the Kerberos Version 5 Authentication Protocol Works".

Host-Based Authentication

Select the **Allow host-based authentication** check box to allow host-based authentication. For more information, see Section 5.8.

Require DNS match

Select the check box to require that the host name given by the client matches the one found in DNS. If the host name does not match, the authentication fails.

Keyboard-Interactive Authentication

Select the **Allow keyboard-interactive authentication** check box to allow keyboard-interactive authentication. For more information, see Section 5.9.

Failure delay / Max tries

Set the delay between failed attempts in seconds (**Failure delay**) and the maximum number of attempts (**Max tries**). The default delay is 2 seconds and default maximum is 3 attempts.

Submethods

For keyboard-interactive authentication, several submethods can be specified.

To edit the submethods, click the **Submethods** button. The **Keyboard-Interactive Submethods** dialog box opens (Figure 4.32).

Password Cache

Select the **Enable Password Cache** check box to enable server password cache. For more information, see Section 4.1.5.

Keyboard-Interactive Submethods

In the **Keyboard-Interactive Submethods** dialog box you can configure the allowed submethods. On Windows, the password, RSA SecurID, RADIUS, and generic submethods are available.

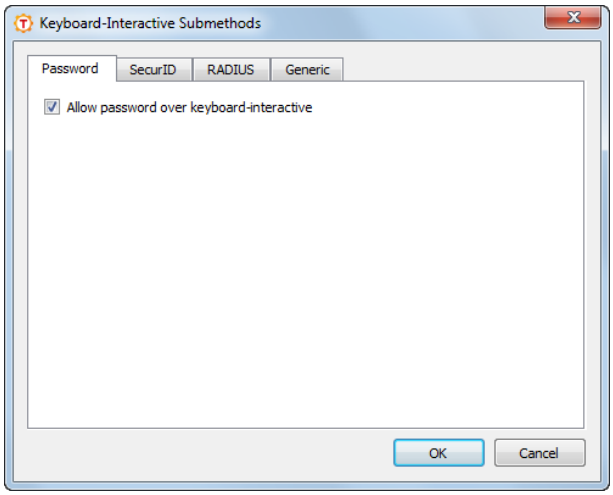


Figure 4.32. Keyboard-interactive submethods

Password

Select the **Allow password over keyboard-interactive** to allow the password submethod. For more information, see Section 5.9.1.

SecurID

Select the **Allow SecurID over keyboard-interactive** to allow the RSA SecurID submethod. For more information, see Section 5.9.3.

DLL Path

Enter the path to the SecurID DLL.

RADIUS

Select the **Allow RADIUS over keyboard-interactive** to allow the RADIUS submethod. For more information, see Section 5.9.4.

Servers

Click **Add** to add a new RADIUS server. The **RADIUS Submethod** dialog box opens.

For each RADIUS server, define a **Shared secret file**, server IP **Address**, **Port**, **Timeout**, and **Client NAS identifier**.

To change the order of the RADIUS servers, select a server from the list, and click **Up** and **Down** to move it. The servers are tried in the specified order.

To edit a RADIUS server, select the server from the list and click **Edit**.

To remove a RADIUS server, select the server from the list and click **Delete**.

Generic

Click **Add** to add a new generic submethod. The **Generic Submethod** dialog box opens.

Enter the **Name** of the method and the initialization **Parameters**.

4.1.13 Services

On the **Services** page you can set restrictions on the services (e.g. terminal, tunneling, SFTP) that the server provides to users.

The selectors define which users a service rule applies to. The order of the rules is important. The first matching rule is used and the remaining rules are ignored.

If no selectors (or only empty selectors) are specified in a service rule, the rule matches all users. The last rule should always be the `(default)` rule, with no selectors. This rule is used for all users that do not match any previous rule. In the simple GUI mode, there is only the default rule and it is used for all connections.

If the user was already put to a services group during authentication (using **Set Services group**), the selectors on the **Services** page are not checked but the corresponding service rule is automatically used.

To add a new service rule, click the **Add** button below the tree view. Each rule will have a sub-page with eight tabs. On the **Selectors** tab, you can edit the selectors of the rule, and on the other tabs, you can configure the allowed services for the rule.

To edit a service rule, select a services item on the tree view. For more information, see the section called "Editing Services Items".

To change the order of the rules, select a services item on the tree view and use the **Up** and **Down** buttons. The rules are read in order, and the first matching service rule on the list is used. The last rule should always be the `(default)` rule, with no selectors.

To delete a service rule, select a services item and click **Delete**.

Editing Services Items

Each item under **Services** has eight tabs, **Selectors**, **Basic**, **SFTP**, **Commands**, **Local Tunnels**, **Remote Tunnels**, **Environment Variables**, and **Subsystems**. Depending on the settings made some of the tabs may be disabled.

Selectors (Advanced Mode)

On the **Selectors** tab, you can configure the selectors that apply to the service rule.

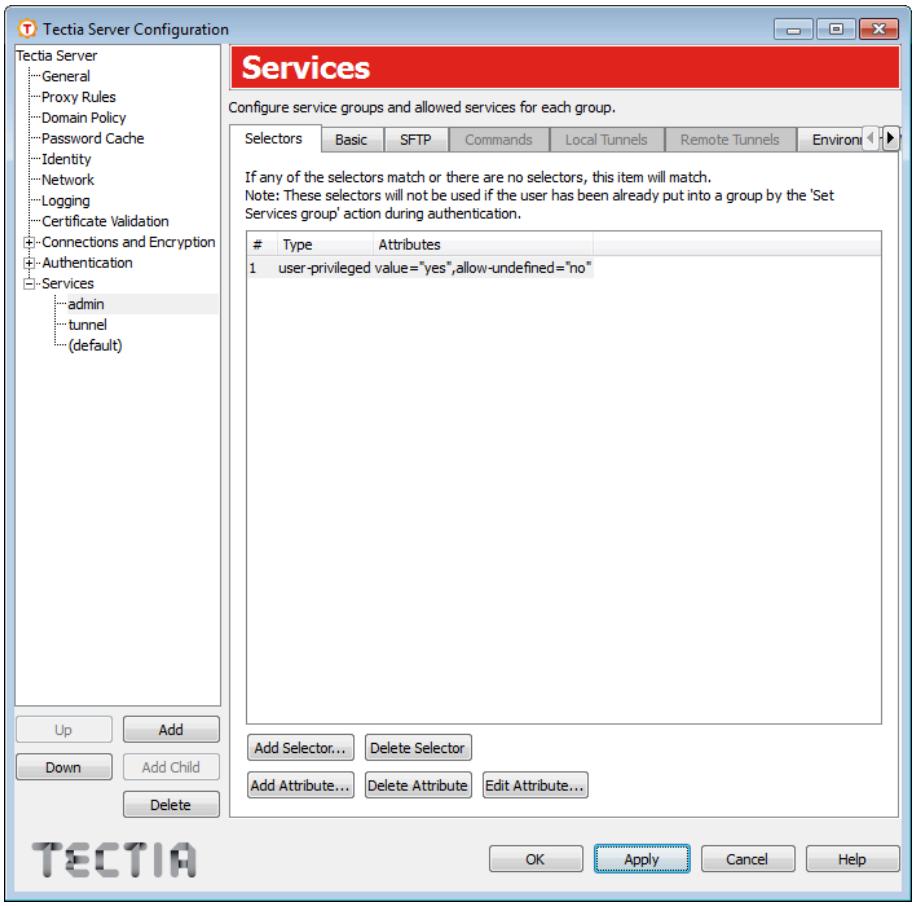


Figure 4.33. Tectia Server Configuration - Services page - Selectors tab

Selector list view

The selector list view shows the selectors that apply to the rule.

To add a new selector to the rule, click **Add Selector**. The new selector will contain automatically at least one attribute. The **Add Selector** dialog box opens allowing you to specify the selector type. For more information on the different selector attributes, see the section called “Editing Selectors”.

To remove a selector, choose the selector from the list view on the **Selectors** tab and click **Delete Selector**. This will delete the selector and all its attributes.

To add a new attribute to a selector, choose a selector from the list and click **Add Attribute**. The **Add Selector** dialog box opens. For more information on the different selector attributes, see the section called “Editing Selectors”.

To edit a selector attribute, choose the attribute from the list and click **Edit Attribute**. The relevant selector dialog box opens. For more information on the different selector attributes, see the section called “Editing Selectors”.

To remove a selector attribute, choose the attribute from the list and click **Delete Attribute**. Note that a selector with no attributes will match everything.

Basic

On the **Basic** tab, you can control the basic settings of the service rule.

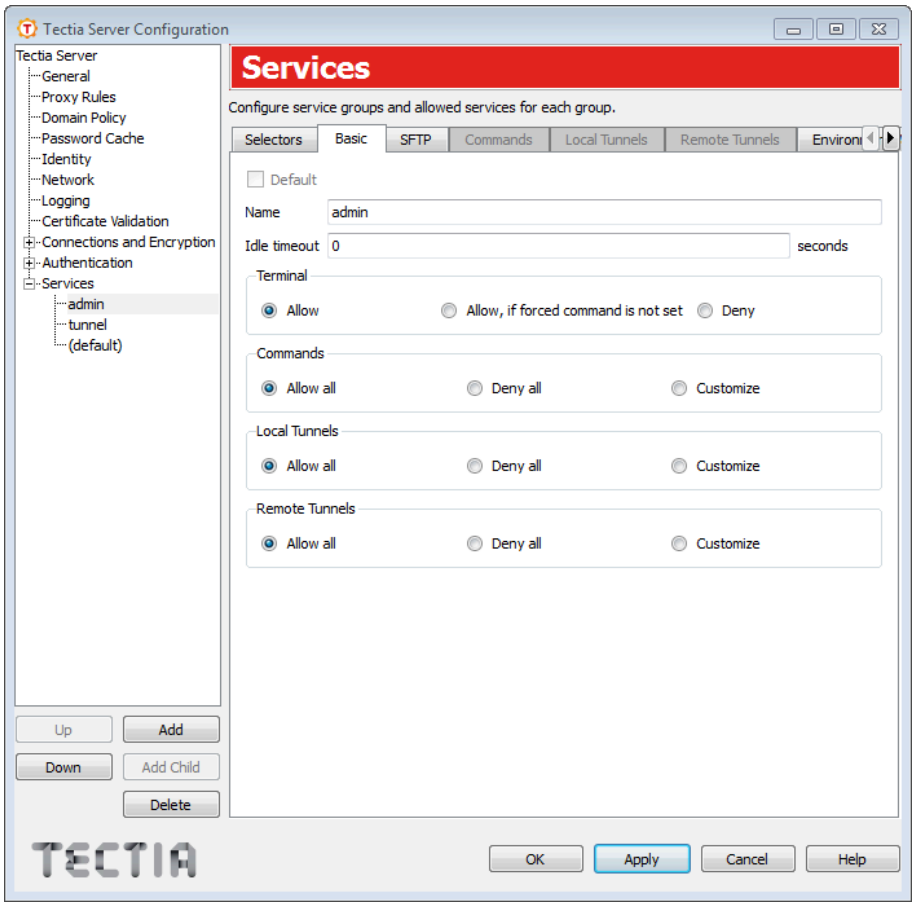


Figure 4.34. Tectia Server Configuration - Services page - Basic tab

To make the rule the default rule, select the **Default** check box. Only one of the rules can be the default rule. The default rule does not have selectors or a name.

The initial default rule allows all users access to all services. The default rule should be kept as the last rule, so it will apply to users that are not matched by any other rule. You should edit the rule according to your security policy.

For other rules than the default, enter the **Name** of the rule.

The **Idle timeout** field sets the idle timeout limit in seconds. If the connection (all channels) has been idle this long, the connection is closed. The default is `0` (zero), which disables idle timeouts.

Terminal

The **Terminal** setting defines whether terminal access is allowed or denied for the users.

To allow terminal in all situations, select **Allow**.

To allow terminal conditionally, select **Allow, if forced command is not set**. If this option is selected and a forced command is defined in the configuration file or in an authorization file, the forced command is run instead of giving the terminal. However, if no forced commands are defined, the user

can get terminal normally. See the section called “Commands” and the section called “Authorization File Options”.

To deny terminal in all situations, select **Deny**.

If terminal access is denied, also shell commands are denied, unless commands are explicitly allowed or set as forced under **Commands**.

Commands

This setting defines whether remote commands are allowed.

To allow all remote commands, select **Allow all**.

To deny all remote commands, select **Deny all**.

To customize remote commands, select **Customize**. You can specify the allowed or forced commands on the **Commands** tab. See the section called “Commands”.

Local Tunnels

This setting defines whether local tunnels are allowed.

To allow all local tunnels, select **Allow all**.

To deny all local tunnels, select **Deny all**.

To customize local tunnels, select **Customize**. You can specify the allowed and denied tunnels on the **Local Tunnels** tab.

Remote Tunnels

This setting defines whether remote tunnels are allowed.

To allow all remote tunnels, select **Allow all**.

To deny all remote tunnels, select **Deny all**.

To customize remote tunnels, select **Customize**. You can specify the allowed and denied tunnels on the **Remote Tunnels** tab.

SFTP

On the **SFTP** tab, you can allow and deny SFTP for users and set limitations on the folders accessible via SFTP and SCP2.

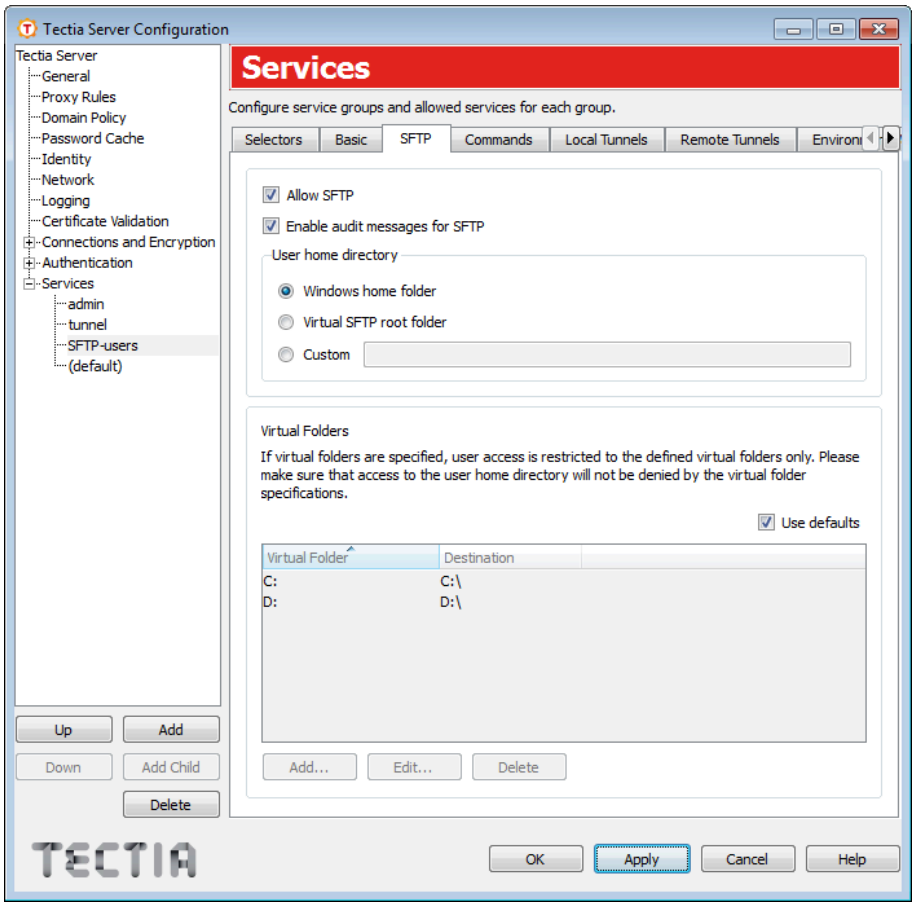


Figure 4.35. Tectia Server Configuration - Services page - SFTP tab

Select the **Allow SFTP** check box to allow SFTP for the users. Clear the check box to deny it.

Note

Denying SFTP denies both SFTP and SCP2 operations to the server, but it does not deny legacy OpenSSH-style SCP operations. To deny OpenSSH SCP (version 8 or older), you should restrict remote commands. See the section called “Basic”.

Enable audit messages for SFTP

This setting defines whether all SFTP server's audit messages are recorded in the system log. By default, all audit messages go to the system log.

On a busy Tectia Server, the system log can grow very rapidly if all audit messages are included in the system log; this configuration option makes it possible to reduce the system log growth rate.

If the check box is cleared, no audit messages from the SFTP server are recorded into the system log.

Enable UTF-8 mode

UTF-8 mode is enabled by default for file names in SFTP server.

If the check box is cleared, the system's code page is used instead for file names on the server side.

User home directory

This setting defines the directory where the user's SFTP session starts and which is the default target for the SCP2 operations (by default **Windows home folder**, `%USERPROFILE%`). The location of the home directory must be under one of the defined virtual folders.

If **Virtual SFTP root folder** is selected (or if a **Custom** directory that is denied by the virtual folder settings is specified), the session will start in the virtual SFTP root folder. See the section called "Defining SFTP Virtual Folders (Windows)" for more information.

Note

The virtual SFTP root folder is not an actual directory on disk and no files can be written there.

Virtual Folders

Virtual folders can be used to restrict the folders the user is able to access via SFTP and SCP2.

If the **Use defaults** check box is selected, all local drive letters are used as defaults. This means that the user can access all drives via SFTP and SCP2.

If any virtual folders are explicitly defined in the configuration, the default drive letters are not used. If you still want to use the drive letters, they need to be defined separately as virtual folders. For more information, see the section called "Defining SFTP Virtual Folders (Windows)".

To define a custom virtual folder:

1. Clear the **Use defaults** check box.
2. Click **Add**. The **SFTP Virtual Folder** dialog box opens.
3. Enter the **Virtual Folder** name.

   To browse for the **Destination**, click the **Browse** button. A **Select Folder** dialog appears, allowing you to specify the desired destination folder. You can also type the path directly into the text field and use special strings as part of the path, for example **%username-without-domain%** (user's login name without the domain part).

   Click **OK**.

Note

Some special characters such as a slash "/" and a backslash "\" cannot be used in the name of the Virtual Folder (for example `network_share`) displayed to the SFTP user.

In case a trailing dollar sign $ is used in the path to the virtual folder (for example `\\server\share$`), the sign has to be escaped as follows:

```
\\server\share$$
```

Figure 4.36 shows an example of a virtual folder setting. The user will see a virtual folder `C:` under the SFTP root folder. When the user changes directory to `C:`, he is actually directed to `C:\SFTP`.

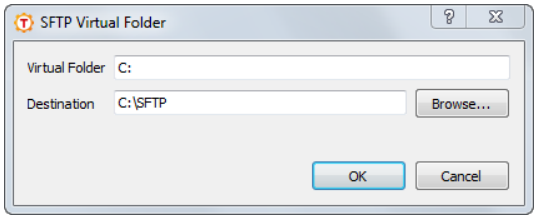


Figure 4.36. Defining a virtual folder

To edit a virtual folder, select the folder from the list and click **Edit**.

To delete a virtual folder, select the folder from the list and click **Delete**.

Note

If you delete all virtual folders, the configuration will revert back to the default settings (local drive letters are available as virtual folders).

Commands

On the **Commands** tab, you can define specific shell commands as allowed or forced. To deny all commands, select **Deny all** on the **Basic** tab.

To add a command rule:

1. Click **Add**. The **Command** dialog box opens.

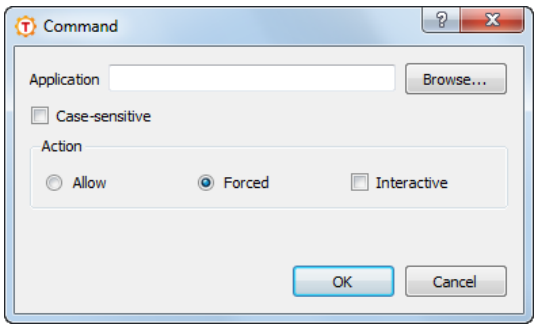


Figure 4.37. Tectia Server Configuration - Services page - Adding commands

2. Select the **Application** and **Action** for the rule.

   If the **Allow** action is set, running the specified application(s) is allowed. All other applications are implicitly denied. Allowed command rules do not apply, if user requests terminal.

If no application is given for the **Allow** action, all commands are allowed. This is equal to selecting **Allow all** on the **Basic** tab.

If the **Forced** action is set, the specified application is run automatically when the user logs in. All other applications are implicitly denied. When you set the **Forced** action, it is possible to set the **Interactive** option on. If the application that is run as forced requires user interaction, set the **Interactive** option on. If the application does not require user interaction, leave the option unchecked. The option is available on Windows only. If you set a forced command, you should also deny terminal. Otherwise, users can request terminal normally, in which case the forced command is not run. Only one forced command per group is allowed. If a forced command is set, no other commands can be added to the service group. If a group contains multiple allowed commands, forced commands can not be added to the group.

Users can also define forced commands for public keys in their `authorization` files. However, if a command is defined in the Tectia Server configuration, it overrides any commands in the authorization files. For more information, see the section called “Authorization File Options”.

Applications are normally matched case-insensitively. Select the **Case-sensitive** check box to match the application case-sensitively.

Click **OK** when finished.

To edit a command rule, select the rule from the list and click **Edit**.

To delete a command rule, select the rule from the list and click **Delete**.

Note

Support for legacy OpenSSH SCP in Tectia Server is implemented using a command called **scp1-compat-srv**. When a client uses OpenSSH version 8 or older SCP to connect to Tectia Server, the server invokes this command. Restrictions on remote commands apply also to OpenSSH-style SCP operations to the server.

Local Tunnels

On the **Local Tunnels** tab, you can define rules for local TCP tunnels (port forwarding). You can add several allow and deny rules with different source address and destination address and port attributes. You can also define an external application to be used to set the tunneling constraints. When a user attempts tunneling, the rules are read in order and the first matching rule is used.

For more information on local tunnels, see Section 8.2.

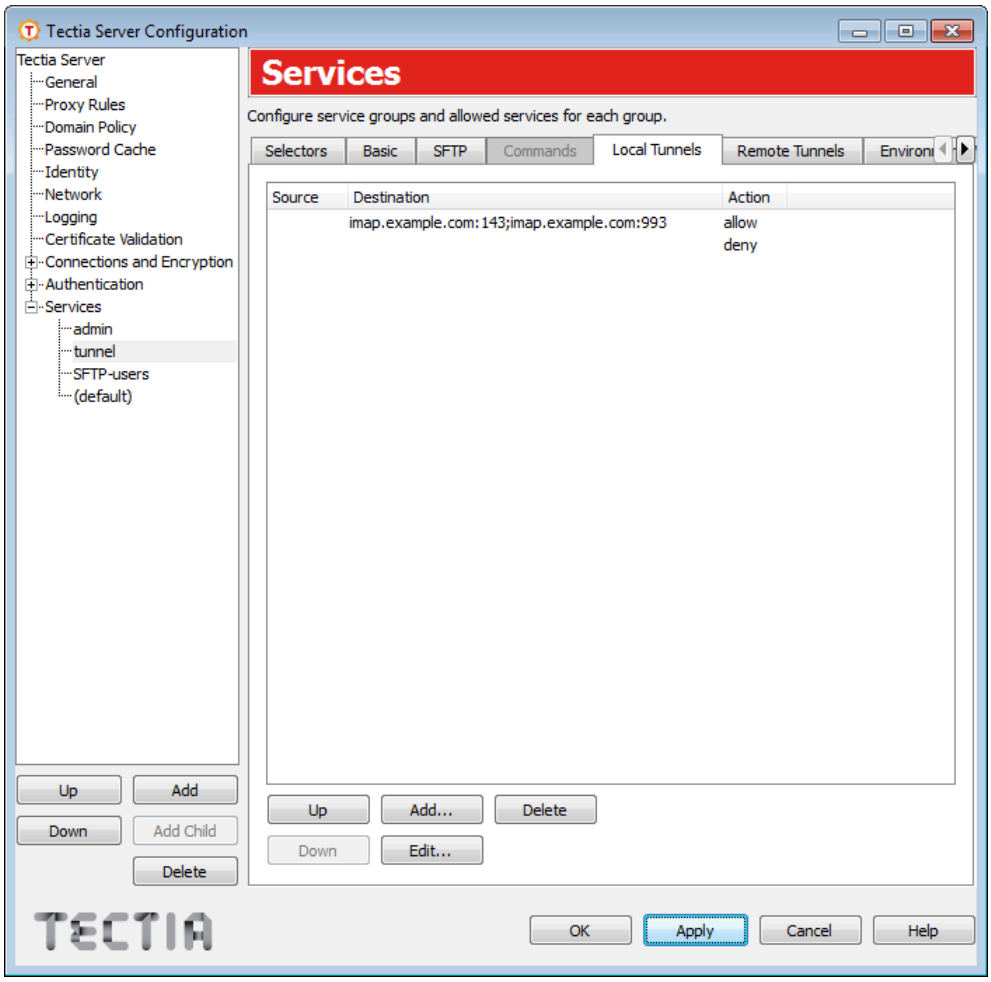


Figure 4.38. Tectia Server Configuration - Services page - Local Tunnels tab

To add a tunneling rule:

1. Click **Add**. The **Local Tunnel** dialog box opens.

2. Select the tunneling **Action** (**Allow** or **Deny**).

   If you define no other settings, the rule will match all tunneling requests.

3. To define additional restrictions for the rule, click **Add**. The **Local Tunnel Definition** dialog box opens.

4. Select whether the definition is for the **Source** or **Destination**, or if you want to use an **External application** to set the tunneling restrictions. Note that for each local tunneling rule, you can define either source(s) and/or destination(s), or an external application; you cannot define an external application in the same rule with source and/or destination definitions.

   The **Address** (IP or FQDN) can be given for the **Source** definition.

   The **Address** (IP or FQDN) and the **Port** can be given for the **Destination** definition.

The **Address** can be in one of the following formats:

- 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 **Fully Qualified Domain Name** can include a comma-separated list of FQDN patterns (case-insensitive). These patterns may also contain "*" and "?" globbing characters. The form of the pattern is not checked.

The **Port** can be either a single port or a port range.

If you want to use an **External application** to set the tunneling restrictions, define the **Command** for executing the application. To select an executable, click the **Browse** button on the right-hand side of the text field. The **Command/Script** dialog appears, allowing you to specify the desired file. You can also type the command directly into the text field.

Caution

The external application will be launched under administrator privileges.

Timeout defines the time limit for the external application to exit. The allowed value range is 1 to 3600 seconds, and the default value is 15 seconds. If the application hangs, Tectia Server will not kill it.

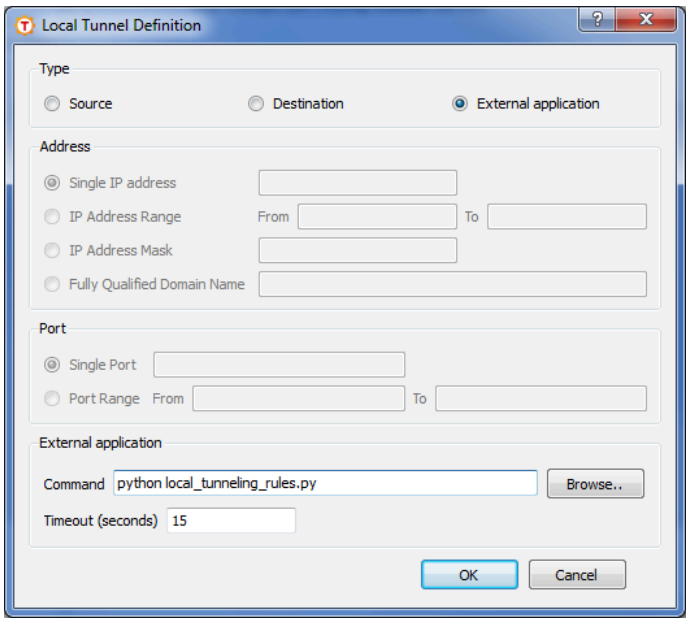


Figure 4.39. The Local Tunnel Definition dialog box

Tectia Server uses the Tectia Mapper protocol (see Appendix E) to communicate with the external application.

Tectia Server sends the following data to the external application:

- `user=userid:username` (specifies the user id and user name)
- `user-privileged=true|false` (specifies whether the user has administrator privileges)
- `{tunnel-src}addr-ip=ip-address` (specifies the tunnel's source IP address)
- `{tunnel-src}port=port` (specifies the tunnel's source port)
- `{tunnel-src}addr-fqdn=FQDN` (specifies the tunnel's source host (fully qualified domain name))
- `{tunnel-dst}addr-ip=ip-address` (specifies the tunnel's destination IP address)
- `{tunnel-dst}port=port` (specifies the tunnel's destination port)
- `{tunnel-dst}addr-fqdn=FQDN` (specifies the tunnel's destination host (fully qualified domain name))

For more information on the communication between Tectia Server and the external application, see Appendix E.

Click **OK** to create the definition and return to the **Local Tunnel** dialog box.

5. You can add one or more Source and/or Destination definitions to each rule, or alternatively one External application definition.

   To edit a definition, select the definition from the list and click **Edit**.

   To delete a definition, select the definition from the list and click **Delete**.

   Click **OK** to create the tunneling rule.

To edit a tunneling rule, select the rule from the list and click **Edit**.

To delete a tunneling rule, select the rule from the list and click **Delete**.

To change the order of the rules, select a rule from the list, and click **Up** and **Down** to move it. The rules are read in order and the first matching rule is used.

Remote Tunnels

On the **Remote Tunnels** tab, you can define rules for remote TCP tunnels (port forwarding). You can add several allow and deny rules with different address and port attributes. When a user attempts tunneling, the rules are read in order and the first matching rule is used.

For more information on remote tunnels, see Section 8.3.

To add a tunneling rule:

1. Click **Add**. The **Remote Tunnel** dialog box opens.

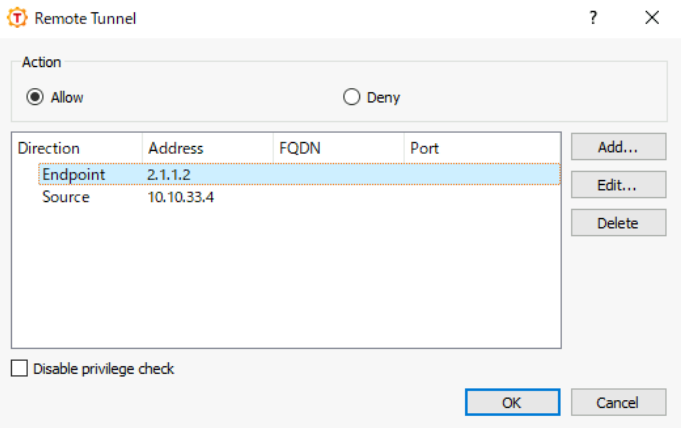


Figure 4.40. The Remote Tunnel dialog box

2. Select the tunneling **Action** (**Allow** or **Deny**).

   If you define no other settings, the rule will match all tunneling requests.

3. To define additional source and listen restrictions for the rule, click **Add**. The **Remote Tunnel Definition** dialog box opens.

4. Select whether the definition is for the **Source** or **Listen**.

   The **Address** (IP or FQDN) can be given for the **Source** definition.

   The **Address** (IP) and the **Port** can be given for the **Listen** definition.

   The **Address** can be in one of the following formats:

   - 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 **Fully Qualified Domain Name** can include a comma-separated list of FQDN patterns (case-insensitive). These patterns may also contain "*" and "?" globbing characters. The form of the pattern is not checked.

   The **Port** can be either a single port or a port range.

   Click **OK** to create the definition and return to the **Remote Tunnel** dialog box.

5. If you want to allow non-privileged users access to privileged ports, enable `Disable privilege check`.

6. You can add several definitions to the rule.

   To edit a definition, select the definition from the list and click **Edit**.

To delete a definition, select the definition from the list and click **Delete**.

Click **OK** to create the tunneling rule.

To edit a tunneling rule, select the rule from the list and click **Edit**.

To delete a tunneling rule, select the rule from the list and click **Delete**.

To change the order of the rules, select a rule from the list, and click **Up** and **Down** to move it. The rules are read in order and the first matching rule is used.

Environment Variables

On the **Environment Variables** tab, you can define the environment variables the users can set on the client side.

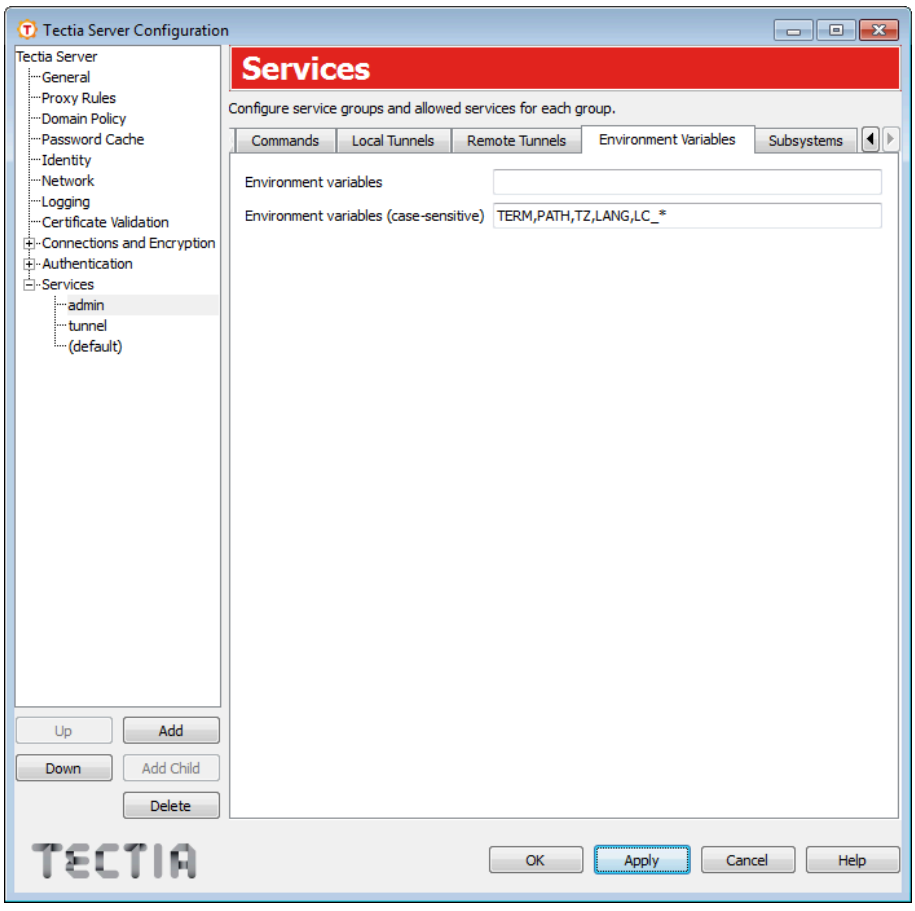


Figure 4.41. The Environment Variables dialog box

To add variables as allowed, enter the variables in the **Environment Variable** field as a comma-separated list.

Allowed variables are normally matched case-insensitively. Enter the variables in the **Environment Variable (case-sensitive)** field to match the variables case-sensitively.

If any variables are set as allowed, all other variables are implicitly denied. Do **not** use * (asterisk), as it will allow any and all variables, and that can be a security risk.

Subsystems

On the **Subsystems** tab, you can define other subsystems (other than SFTP) as allowed or denied. The most commonly used subsystem, SFTP, can be allowed and denied directly from the **SFTP** tab.

To add a subsystem, click **Add**. Enter the subsystem **Type** and select whether to **Allow** or **Deny** the subsystem. Define also the **Application** which is the the executable of the subsystem.

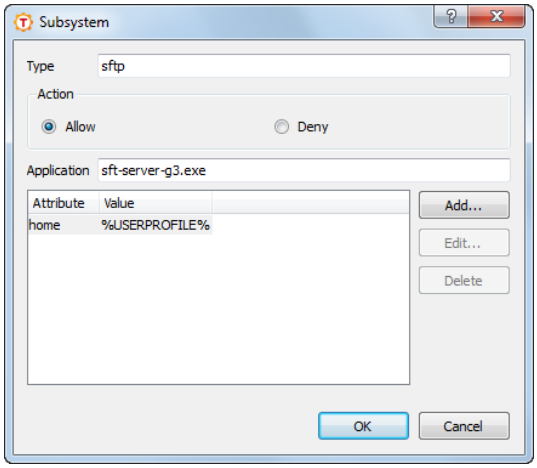


Figure 4.42. Adding a new subsystem dialog box

The subsystem can contain several attributes. To add an attribute, click **Add**. Enter the **Attribute** and its **Value** and click **OK**.

The attributes can be used, for example, on Windows platforms to set the user home directory and virtual folders for SFTP, as in the example screen above.

To edit an attribute, select an attribute and click **Edit**.

To remove an attribute, select an attribute and click **Delete**.

4.2 Configuration File for Tectia Server

This section introduces the XML-based Tectia Server configuration file `ssh-server-config.xml`, its structure, elements and options. For descriptions of the elements, see ssh-server-config(5). For a quick reference to the elements and their attributes, see Appendix A. For information on the syntax of the configuration file, see Appendix B.

The configuration file follows the same logic that is used in setting up a Secure Shell connection:

1. The settings related to the server's own configuration are made in the `params` block.

2. After the client has initiated a connection to the server, the server checks whether connections from the client address are allowed. The client and server perform key exchange where the server authenticates itself to the client, and ciphers, KEXs and MACs are selected for the connection. The related configuration settings are made in the `connections` block.

3. The server requests the user to authenticate itself to the server. The server may offer a selection of authentication methods, or require several authentication methods to be passed in succession. The configuration settings related to authentication are made in the `authentication-methods` block.

4. The server determines the services the client is allowed to use. The related configuration settings are made in the `services` block.

Note

The configuration file is read in top-down order during connection setup. If a connection is denied in one of the blocks, the connection setup phase ends immediately and the rest of the configuration file will not be read at all.

4.2.1 Dividing the Configuration into Several Files

It is possible to divide the Tectia Server configuration into several files. You can define external XML files containing sub-configurations for example with department-specific or user-group-specific settings. This can make the configuration easier to manage as it is in smaller parts, and the sub-configuration files can be used repeatedly in several places.

Note

Configuration files consisting of several XML-files must be maintained manually, because split configuration files cannot be edited with the Tectia Server Configuration GUI.

The sub-configuration files must be declared as external SYSTEM entities within the DOCTYPE element of the `ssh-server-config.xml` file. For example the `entity-name` below:

```
<!DOCTYPE secsh-server SYSTEM
   "/opt/tectia/share/auxdata/ssh-server-ng/ssh-server-ng-config-1.dtd" [

   <!ENTITY entity-name SYSTEM "sub-config-file.xml">
```

The defined entity can then be used in the main configuration file instead of defining all the settings there. The server configuration will read the contents of the sub-configuration file in the place of the entity. So the sub-configuration file contents must be designed so that they produce a valid XML structure in the `ssh-server-config.xml` file.

In this example we have a sub-configuration file named `group-example-rules.xml`, located in sub-directory `subconfigs/`, and with the following contents:

```
<terminal action="deny" />
<subsystem type="sftp" application="sft-server-g3" chroot="%homedir%" />
<tunnel-agent action="deny" />
```

```
<tunnel-x11 action="deny" />
<tunnel-local action="deny" />
<tunnel-remote action="deny" />
```

In the example below, we first declare the sub-configuration file (and its location) as an external entity in the beginning of the `ssh-server-config.xml` file, and then use the `group-A-rules` entity in the actual configuration as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE secsh-server SYSTEM
   "/opt/tectia/share/auxdata/ssh-server-ng/ssh-server-ng-config-1.dtd" [

  <!ENTITY group-A-rules SYSTEM "subconfigs/group-example-rules.xml">
]>
<secsh-server>
  ...
  <services>
    <group name="example">
      <selector>
         <user-group name="example"/>
      </selector>
    </group>
     ...
     <rule group="example">
      &group-A-rules;
    </rule>
     ...
  </services>
</secsh-server>
```

4.2.2 Using Selectors in Configuration File

The connection settings can be changed based on *selectors* in the configuration file. Using selectors makes it possible, for example, to:

- allow/deny connections from certain IP addresses
- require different authentication methods based on user name or group
- restrict access based on a certificate field

Selectors are used in the `connections`, `authentication-methods`, and `services` blocks.

The different selector attributes are specified as sub-elements of the `selector` element. The following sub-elements are available:

- `certificate`: Matches to a pattern in a specified field of the user certificate.
- `host-certificate`: Matches to a pattern in a specified field of the client host certificate.
- `interface`: Matches to the server listener interface ID or address/port.
- `ip`: Matches to the IP address or FQDN of the client.

- `publickey-passed`: Matches if the authentication is passed using a normal public key without a certificate.
- `user`: Matches to a user name or ID.
- `user-group`: Matches to a user group name or ID.
- `user-password-change-needed` (*Unix*): Matches if the user password has expired and should be changed.
- `user-privileged`: Matches based on the user privilege status (yes/no).
- `blackboard`: Matches based on information stored on the blackboard, for example, channel codes, authentication methods. For more information, see **blackboard** .

In the `connections` block, only the `interface` and `ip` selectors can be used. In the `authentication-methods` and `services` blocks, all selectors can be used.

When a parent element contains multiple child elements with selectors, the first functional child element that matches will be used, and the rest will be ignored. Note that because of this, if the `connections` element has multiple `connection` child elements, but the first one has an empty selector, or no selectors at all, that `connection` element will always match and the remaining ones will never be used.

Note that in Windows domain environment, the `user` and `user-group` selectors have a length limitation. For more information, see the description of option **User** in Section 4.1.

Wildcards in Selectors

Simple wildcards can be used in the `user` or `user-group` selector values:

- An asterisk (*****) matches any number of any characters.
- A question mark (**?**) matches any single character.
- A hyphen (**-**) can be used in the `user id` and `user-group id` values to match a range of integers.

For example, the following selector matches to user names `jdoe` and `jdox`, but not to `jdoex`:

```
<selector>
  <user name="jdo?" />
</selector>
```

Regular Expressions in Selectors

Regular expressions can be used in selectors to define ranges of values instead of defining each possible value separately. Each regular expression attribute (`regexp`, `fqdn-regexp` or `name-regexp`) always contains a single pattern, never lists of patterns. The whole string must be matched for a match to be successful.

Regular expressions are applicable for example with user names and user group names. For example, the following selector matches to all user names that consist of (any) 4 letters and (any) 3 numbers:

```
<selector>
```

```
  <user name-regexp="[[:alpha:]]{4}[[:digit:]]{3}" />
</selector>
```

By default, the regular expressions are matched case-insensitively. Case-sensitive matching can be activated by adding `"(?-i)"` to the pattern. This instructs the regexp engine to turn off case-insensitive matching for the following string. It can be turned back on with `"(?i)"`.

Note

Design the regular expressions very carefully in order to avoid unintentional matches.

For the syntax of the regular expressions, refer to the description of the Egrep Syntax in the *Tectia Client User Manual* or *Tectia ConnectSecure Administrator Manual*. Do not use the control characters elsewhere in the values, but if it is unavoidable, carefully escape the relevant characters. The escape character is backslash "\". A literal backslash can be matched with "\\".

Selector Processing

Multiple `selector` elements are in an OR relation (one of the selector elements must match for the parent element to match). For example, the following block matches if either the IP address is *`192.168.0.3`* or the user ID is *`1001`*:

```
<selector>
  <ip address="192.168.0.3" />
</selector>
<selector>
  <user id="1001" />
</selector>
```

Selector attributes in the same `selector` element are normally in an AND relation (all attributes must match for the element to match). For example, the following block matches if both the IP address is *`192.168.0.3`* and the user ID is *`1001`*:

```
<selector>
  <ip address="192.168.0.3" />
  <user id="1001" />
</selector>
```

However, selector attributes in the same `selector` element matching to the same attribute type are in an OR relation to each other. The following three examples produce the same result, either the user name `exa` or `mple` matches:

```
<selector>
  <user name="exa" />
  <user name="mple" />
</selector>

<selector>
  <user name="exa" />
</selector>
<selector>
  <user name="mple" />
```

```
</selector>

<selector>
  <user name="exa,mple" />
</selector>
```

An empty selector always matches:

```
<selector />
```

Also, typically, if an element accepts selectors, but none are given, the element is assumed to have an empty selector, which will then always match.

Selectors and Undefined Data

Normally when the server tries to match to a selector attribute for which the respective data has not been defined (the data is not available to the server), the selector matching process ends in error, effectively terminating the connection attempt. This happens, for example, in the following cases:

- Other selectors than `ip` and `interface` are erroneously used in the `connections` block. Only the IP address of the client and the connected listener interface are available to the server in that stage of connection. For example, the user name is not yet known.
- The `certificate` selector is erroneously used without previously requiring public-key authentication. The server will not have user certificate data unless it has received it first during public-key authentication.
- The `host-certificate` selector is erroneously used without previously requiring host-based authentication. The server will not have host certificate data unless it has received it first during host-based authentication.
- The `certificate` or `host-certificate` selector is used to match to a field that does not exist in the certificate.
- The `user-privileged` selector is used in the `authentication-methods` block on a Windows server and the user is logging in using a domain account and does not yet have an access token allocated.

The `allow-undefined` attribute can be used in all selector sub-elements to control this behavior. Its value must be `yes` or `no`. If set to `yes`, the undefined data is treated as non-matched and the matching continues to other elements. The default is `no` (trying to match undefined data results in termination of the connection).

For example, encountering the following selector causes the connection attempt to end in failure if the certificate is not available or does not contain the `altname-email` field:

```
<selector>
  <certificate
   field="altname-email"
   pattern="%username%@ssh.com" />
</selector>
```

The following selector simply does not match when the certificate does not exist or does not contain the `altname-email` field, and the processing continues with the next block:

```
<selector>
  <certificate
   field="altname-email"
   pattern="%username%@ssh.com"
   allow-undefined="yes" />
</selector>
```

4.2.3 ssh-server-config.xml

This section describes the elements and options available in the XML-based Tectia Server configuration file, ssh-server-config(5).

ssh-server-config

ssh-server-config — Tectia Server configuration file format

The Tectia Server configuration file `ssh-server-config.xml` is a valid XML file.

On Unix, the configuration related files are stored in the following directories:

- `/etc/ssh2/` contains the `ssh-server-config.xml` file
- `/opt/tectia/share/auxdata/ssh-server-ng` contains the XML DTD.

Note

In Tectia Server 6.1 and earlier on Unix the default auxiliary data directory `auxdata` was located in `/etc/ssh2/ssh-tectia/`. If your `ssh-server-config.xml` file was created for Tectia Server version 6.1 or earlier, please update its DOCTYPE declaration to contain the current path to the server configuration file DTD directory: `/opt/tectia/share/auxdata/ssh-server-ng/`.

On Windows, the configuration related files are stored in the following directories:

- "`<INSTALLDIR>\SSH Tectia Server`" contains the `ssh-server-config.xml` file
- "`<INSTALLDIR>\SSH Tectia AUX\ssh-server-ng`" contains the XML DTD.

If the configuration file cannot be found or some of the elements are missing, hardcoded default values are used. You can view the default values in the `ssh-server-config-default.xml` file that is stored in the same directory with the configuration file.

The `ssh-server-config.xml` configuration file is divided into four blocks:

- General server parameters (`params`)
- Connection rules and encryption methods (`connections`)
- Authentication rules and methods (`authentication-methods`)
- Service rules (`services`)

In the `connections` and `authentication-methods` blocks, different `selectors` can be used to set access rules to users based on the user parameters such as user name or location. Users can be divided to groups dynamically, for example, based on the authentication method they use to log in. In the `services` block, each group can then be allowed or denied services such as tunneling, file transfer, and terminal access.

Document Type Declaration and the Root Element

The server configuration file is a valid XML file and starts with the Document Type Declaration (DTD) inside the `DOCTYPE` element. Both the `DOCTYPE` declaration and the DTD are mandatory; should they be missing, the server will not be able to parse the configuration properly.

The root element in the configuration file is `secsh-server`. It can include `params`, `connections`, `authentication-methods`, and `services` elements. These elements in turn can include more elements according to the configuration file syntax, see Appendix B.

An example of an empty configuration file is shown below:

```
<!DOCTYPE secsh-server SYSTEM
  "/opt/tectia/share/auxdata/ssh-server-ng/ssh-server-ng-config-1.dtd">

<secsh-server>

  <params />
  <connections>
    <connection />
  </connections>
  <authentication-methods />
  <services>
    <rule />
  </services>

</secsh-server>
```

Note

It is not mandatory to include all elements in the configuration file. If an element is missing, the equivalent default values shown in the `ssh-server-config-default.xml` file will be used.

The `params` Block

The `params` block defines the general server parameters, such as the location of the host key file, the listen address, logging, connection limits, and certificate validation settings.

address-family

This element defines the network address family used for connections. If `type` is set to `inet`, the server will accept only IPv4 incoming connections. If set to `inet6`, the server will accept only IPv6 incoming connections. If set to `any`, the server will accept both IPv4 and IPv6 incoming connections, will resolve addresses of both families, and opens both IPv4 and IPv6 listeners for remote port forwarding.

```
<address-family type="inet|inet6|any" />
```

The default is `inet`. Command-line options override settings from configuration file.

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`) crypto library can be used. The library name is given as a value of the `mode` attribute. By default, standard crypto libraries are used. The OpenSSL cryptographic library is used in the *FIPS mode*.

```
<crypto-lib mode="fips" />
```

Note

Tectia Server has to be restarted after changing the FIPS mode setting. Extra checks are done when starting the Tectia Server and Connection Broker in the FIPS mode due to the OpenSSL FIPS crypto library health check. This will lead to a noticeable delay in the start of the process on slow machines.

Note

You can switch all Tectia products to FIPS mode before or after installation by creating a file named `FIPSMODE` in the following location:

- On Unix: `/etc/ssh2/FIPSMODE`
- On Windows: `FIPSMODE` file in the `SSH Tectia AUX` folder. This file is created and removed automatically when FIPS mode is changed with the Tectia Server Configuration GUI and configuration is applied.

On Linux and Solaris you can enable and disable `FIPSMODE` file by running the following commands respectively:

```
# /opt/tectia/sbin/ssh-modeset fips-mode on
```

```
# /opt/tectia/sbin/ssh-modeset fips-mode off
```

You may then verify your current FIPS mode with:

```
# /opt/tectia/sbin/ssh-modeset fips-mode-check
```

Caution

To be FIPS compliant you need to regenerate and rotate server's hostkey (and any user authentication keys) if they have not been generated in FIPS mode. Unix installer writes logs to existing FIPSMODE file if hostkey is generated in FIPS mode. On Linux, this happens also if operating system has FIPS enabled. To replace (without rotating) the existing default hostkey with a passphrase protected key:

```
# ssh-keygen-g3 -H --fips-mode --random-pass
```

For more information about key rotation, see Section 5.2.3.

For more information on the functions used from the cryptographic library, see **Cryptographic library**.

settings

This element contains miscellaneous settings. It has the following attributes: `proxy-scheme`, `xauth-path`, `xauth-shell`, `x11-listen-address`, `pam-account-checking-only`, `resolve-client-hostname`, `ignore-aix-rlogin`, `ignore-aix-login`, `record-ptyless-sessions`, `user-config-dir`, `default-path`, `windows-logon-type`, `windows-terminal-mode`, `ignore-nisplus-no-permission`, `quiet-login`, `terminate-user-processes`, and `allow-elevation`.

The `proxy-scheme` attribute defines rules for HTTP or SOCKS proxy servers that Tectia Server uses when a client forwards a connection (local tunnel).

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 host name which may be a regular expression containing the characters "*" and "?" and a port range:

```
name_pattern[:port_range]
```

An example `proxy-scheme` 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.

```
"direct:///127.0.0.0/8,*.ssh.com;
 http-connect://http-proxy.ssh.com:8080/*.example;
 socks://fw.ssh.com:1080/"
```

The `xauth-path` attribute contains a path to a supplementary XAuth binary used with X11 forwarding on Unix platforms.

The `xauth-shell` attribute specifies the shell used to run xauth binary. The default is to use the user shell.

On Unix, the `x11-listen-address` attribute can be used to configure on what kind of address the x11 listener (used in X11 forwarding) is created. Possible values are:

- `localhost` (default) - sets the DISPLAY environment variable to `127.0.0.1:<screen>`, where `<screen>` is the tunneled screen number, typically 10.0. This means that the x11 listener is bound to a loopback address; this setting should be sufficient for most use cases.
- `any` - sets the DISPLAY environment variable to `<address:screen>`, where `<address>` is the interface to which the SSH session is bound (typically the first network interface) and the `<screen>` is the tunneled screen number, typically 10.0. This setting will bind the X11 listener to the 0.0.0.0 (wildcard) interface thereby allowing connections to the proxy from other hosts. Use this setting on HPUX systems, if you need to tunnel older X11 applications (such as hpterm).

  When `x11-listen-address=any`, the SO_REUSEADDR socket option will be left non-set in order to prevent the possibility of session hijacking on some operating systems by other users binding to the same port with a more specific address.

On Unix, the `pam-account-checking-only` attribute can be used to define that only PAM will be used to check if the user is allowed to login (for example, the account is not locked). Generally, PAM can be used during authentication or in PAM account or session management via the `pam-calls-with-commands` setting.

Possible values for the `pam-account-checking-only` attribute are:

- `yes` - only PAM is used to check the user account and Tectia Server will not try to independently verify whether the account has been locked or otherwise disabled, if either PAM authentication has succeeded or if `pam-calls-with-commands` is set.
- `no` (default) - the normal system checks will be used to determine whether the user is allowed to login, regardless of the result of PAM authentication or the `pam-calls-with-commands` setting.

The `resolve-client-hostname` attribute can be used to define whether Tectia Server should try to resolve the client host name from the client IP address during connection setup.

If an IP address of the client host is defined with the `Allow/Deny-from` option in the authorization file, then the `resolve-client-hostname` attribute is ignored. But if a host name is defined with the `Allow/Deny-from` option, then this attribute is used.

Note

This attribute does not affect the resolution of TCP tunnel endpoints and Tectia Server will try to resolve the client host name when creating a TCP tunnel.

Possible values for the `resolve-client-hostname` attribute are:

- `yes` (default) - DNS lookups are used to resolve the client host name at connection time
- `no` - client host name resolution is not attempted, but the IP address is used as the returned client host name. This is useful when you know that the DNS cannot be reached, and the query would cause just additional delay in logging in.

The `ignore-aix-rlogin` attribute defines whether the server should ignore the **remote login** restriction on AIX. Possible values are:

- `yes` - Tectia Server will ignore these operating system settings:
  - the rlogin restriction flag
  - the unable to login at this time flags (e.g. logintimes)
- `no` (default)

The `ignore-aix-login` attribute defines whether the server should ignore the **local login** restriction on AIX. Possible values are:

- `yes` - Tectia Server will ignore these operating system settings:
  - the login restriction flag
  - the unable to login at this time flags (e.g. logintimes)
- `no` (default)

The `record-ptyless-sessions` attribute can be used to control whether sessions without PTYs are recorded as user logins in the operating system. Sessions without PTYs are for example remote commands and SFTP sessions. By default, all sessions are recorded. However, some system utilities (such as finger on Solaris) do not allow sessions without PTYs to be recorded because these sessions do not have a valid TTY name. On these systems, only real shell logins should be recorded and others turned off by setting `record-ptyless-sessions=no`. The value must be `yes` or `no`. The default is `yes`.

The `user-config-dir` attribute can be used to specify a directory where user-specific configuration data is to be found, if the data is not stored in the default location. With this setting, the administrator can control those options that are usually controlled by the user. Tectia Server expects Tectia-style directory structure under the given directory, for example, the `/authorized_keys` directory, and the `/authorization` file, if they are being used.

For `user-config-dir`, the default is `%D/.ssh2`. The directory path can include pattern strings which will be expanded by Tectia Server. The following pattern strings can be used:

- `%D` or `%homedir%` is the user's home directory
- `%U` or `%username%` is the user's login name

  For Windows domain users:

  - `%U` is expanded to `domain.username`
  - `%username%` is expanded to `domain\username`

  For local server machine users:

  - `%U` is expanded to `username`
  - `%username%` is expanded to `username` (without the domain prefix)

- `%IU` or `%userid%` is the user's user ID (uid) (*Unix only*)
- `%IG` or `%groupid%` is the user's group ID (gid) (*Unix only*)
- `%installdir%` is the installation directory

On Unix, the `default-path` attribute can be used to define the default PATH value for the user environment. This path will be applied after connection to a server unless anything else is defined in the system settings. Alternatively, the default environment can be set by using the environment variable `PATH`.

On Windows, the `windows-logon-type` attribute can be used to define what kind of user logon methods for the local host are accepted by Tectia Server. The defined logon type affects password authentication. This attribute takes values `batch`, `interactive`, `network`, and `network-cleartext`. The default value is `interactive`.

For example, to enable accounts that do not have the access right to log on locally, make the following setting:

```
<settings windows-logon-type="network" />
```

For information on the attribute values, refer to Microsoft documentation on the Windows logon types.

On Windows, the `windows-terminal-mode` attribute can be used to define the mode of operation of a terminal session on the server side. This attribute takes values `console` and `stream`.

If set to `console` (default), the server reads the screen buffer in a loop and detects modifications based on current cursor location. If set to `stream`, the server reads the stdout and stderr of **cmd.exe** as a stream of data, while providing basic facilities for command-line editing.

On Linux and Solaris, the `ignore-nisplus-no-permission` attribute can be used to define whether Tectia Server should ignore it if NIS+ gives no permission to the user during authentication. The value can be `yes` or `no`. The default is `no`.

When set to `yes`, and when the user to authenticate is not root, the server will ignore it if the NIS + returns *NP* when querying for shadow password. *NP* indicates no permission to read the password information.

Note

When NIS+ returns *NP*, the user will NOT be able to use password authentication or the keyboard-interactive with password authentication to authenticate the session. However, the keyboard-interactive with PAM is possible.

The `quiet-login` attribute can be used to define whether to suppress the messages about last login, password expiry, messages of the day and other such information during login. This can be useful in case third party applications are used to launch a shell, and the messages are useless or even confusing to the applications. The attribute value can be `yes` or `no`. The default is `no`.

The `default-domain` attribute can be used to append a domain to server host names that are not fully qualified domain names (FQDN). For example, if the host name is `server01` and you define:

```
<settings default-domain="example.com"/>
```

then the resulting FQDN will be `server01.example.com`.

When `terminate-user-processes` is enabled, users' processes are automatically terminated upon session end. The attribute value can be `yes` or `no`. The default is `no`.

When `allow-elevation` is enabled, users may retain any administrator privileges associated with their account by appending `elevated,` to their username. For example:

```
$ sshg3 elevated,Administrator@example.com
```

Note that `allow-elevation` only affects password logins.

Example of the `settings` element and its attributes:

```
<settings
 proxy-scheme="direct:///10.0.0.0/8,localhost;socks5://fw.example.com:1080/"
 xauth-path="/usr/X11R6/bin/xauth"
 ignore-aix-rlogin="no"
 ignore-aix-login="no"
 record-ptyless-sessions="yes"
 user-config-dir="%D/.ssh2"
 default-path=""
 windows-logon-type="interactive"
 windows-terminal-mode="console"
 ignore-nisplus-no-permission="no"
 default-domain="example.com"
 quiet-login="no"
 terminate-user-processes="no"
 allow-elevation="no" />
```

pluggable-authentication-modules

This element can be used to define defaults for PAM account management and session management with the following attributes:

- The `pam-calls-with-commands` attribute defines whether PAM Account Management (pam_acct_mgmt) and PAM Session Management (pam_session_mgmt) are enabled when the user executes shells, remote commands and subsystems. The values are `yes` and `no`. The default is `no`, which disables PAM session and account management. This setting has no effect on platforms which do not support PAM.

  Enabling `pam-calls-with-commands` will enforce the PAM restrictions on session and account management regardless of the authentication method that is used to connect to the server. Note that this requires either a PAM configuration file for the service `ssh-server-g3` or the use of the `service-name` attribute to specify the service used by PAM.

- The `service-name` attribute can be used to instruct PAM about which configuration it should use. When defined, this setting will override the factory setting which is `ssh-server-g3`. Note that it

is possible to define different service names for user session management and for authentication by defining different values for the `service-name` attribute in the `pluggable-authentication-modules` element and in the `submethod-pam` element.

- Attribute `dll-path` can be used to define the location of the PAM library, if the library is not in the default library path of the operating system.

  If the PAM library is not in the default library path then the `dll-path` attribute is needed both here in `pluggable-authentication-modules` and in the `authentication-methods` setting `submethod-pam`.

The settings made in `pluggable-authentication-modules` can be overriden locally by adding PAM-related settings also to the `authentication-methods` block. Under the `auth-keyboard-interactive` element you can define the `submethod-pam` element with attributes `service-name` and `dll-path`.

With the following example configuration, the factory settings are overriden by defining `sshd2` as the service for PAM (in the `pluggable-authentication-modules` setting). This service will be used by `ssh-user-exec`, for example. To make the PAM submethod to use the `ssh-server-g3` service, it is specified with the `service-name` in the `submethod-pam` setting.

```
<params>
 <pluggable-authentication-modules service-name="sshd2"/>
</params>
        <!-- ... -->
<authentication-methods>
  <authentication action="allow" name="allow-default">
    <auth-keyboard-interactive>
      <submethod-pam service-name="ssh-server-g3" />
    </auth-keyboard-interactive>
  </authentication>
</authentication-methods>
```

protocol-parameters

This element contains protocol-specific values that can be used to tune the performance. It should be used only in very specific environments. In normal situations the default values should be used.

The `threads` attribute can be used to define the number of threads the protocol library uses (fast path dispatcher threads). This attribute can be used to allow more concurrent cryptographic transforms in the protocol on systems with more than four CPUs. If the value is set to zero, the default value is used.

Example of the `threads` attribute:

```
<protocol-parameters threads="8" />
```

hostkey

This element defines the location of the private host key and optionally the location of the public key and/or certificate. The elements inside the element must be given in the right order (private key before public).

The following attributes can be used to define aspects of host key rotation:

- The `status` attribute defines whether the key file will be used as a host key. The possible values are `normal` and `disabled`. A disabled key can be advertised, even as it is not used as a host key.
- The `advertise` attribute defines whether the key will be advertised. The possible values are `no`, `yes`, and `tectia-only`. When the value is `tectia-only`, the key will be advertised only to Tectia clients.
- The `rotation-period` attribute defines the time span between key rotations. The value is entered in seconds by default, but it can be modified with the `m`, `minutes` , `h`, `hours`, `d`, and `days` suffices to represent those time periods. Rotation period is counted from the current host key timestamp.
- The `rotation-margin` attribute defines a time span before the key rotation. At the start of the margin period, a new host key is generated, and advertisement of the new host key starts.

For more information about key rotation, see Section 5.2.3.

Inside one `hostkey` element either the public key or the certificate can be given, not both.

Giving the public key in the configuration file is not mandatory. It will be derived from the private key if it is not found otherwise. However, specifying the public key will decrease the start-up time for the software, as deriving the public key is a fairly slow operation.

private

The `private` element gives the path to the private key file as a value of the `file` attribute.

The key file should be located on a local drive. Network or mapped drives should not be used, as the server program may not have proper access rights for them. The default is `hostkey`, in the `/etc/ssh2` directory on Unix and in the "`<INSTALLDIR>\SSH Tectia Server`" directory on Windows.

On Unix, the private key file should be readable and writable only by `root`. The private key directory should be writable only by `root`.

On Windows, the key file and directory should have full permissions for the **Administrators** group and the **SYSTEM** account and no other permissions.

public

This element gives the path to the public key file as a value of the `file` attribute.

The key file should be located on a local drive. Network or mapped drives should not be used, as the server program may not have proper access rights for them.

Alternatively, the public key can be specified as a base64-encoded ASCII element.

openssh-certificate

This element gives the path to the OpenSSH host certificate file as a value of the `file` attribute.

x509-certificate

This element gives the path to the X.509 user certificate file as a value of the `file` attribute.

Alternatively, the certificate can be specified as a base64-encoded ASCII element.

externalkey

This element defines an external host key. The `type` must be given as an attribute. The currently supported types are `none`, `software`, `mscapi`, `pkcs11`, and `pkcs12`. The `init-info` for the external key can also be given.

Sample `hostkey` elements are shown below:

```
<hostkey>
  <private file="/etc/ssh2/hostkey_rsa" />
  <public file="/etc/ssh2/hostkey_rsa.pub" />
</hostkey>

<hostkey>
  <private file="/etc/ssh2/hostcert_rsa" />
  <x509-certificate file="/etc/ssh2/hostcert_rsa.crt" />
</hostkey>
```

For PKCS#11, the `<hostkey>` settings are as follows:

```
<hostkey>
  <externalkey type="pkcs11"
               init-info="dll(C:\PKCS11\lib\provider.dll)
                          slots(all)
                          passphrase_file(C:\path\pkcs11-pin)" />
</hostkey>
```

In the PKCS#11 example `<hostkey>` setting, the PKCS `provider.dll` is loaded and initialized from the specified dynamic library path and all available keys and certificates from `slots` are added as server's identity. Typically the provider requires a PIN that can be saved to a separate file `pkcs11-pin` that is specified with the `passphrase_file` in the initialization info.

In the `init-info` string, the following keywords are supported for the `pkcs11` type:

- `dll(<pkcs#11 driver path>)` - defines the full path to dynamic library of the PKCS provider. Both key and optional certificate available from the provider are added to `sshexternalkey`.
- `passphrase(<token passphrase>)` - if the provider requires a PIN or passphrase to access the HSM (Hardware Security Module), the configured passphrase is used. In case the passphrase is not valid, the server fails to start with "Key requires passphrase" error.
- `passphrase_file(<token passphrase filename>)` - defines that instead of giving the passphrase or PIN in the configuration file directly, it can be written to a separate file. This option is useful if server configuration file needs to be more widely readable. The passphrase or PIN can still be with admin access only.

- `slots(<slots filter>)` - defines a specific slot or 'all' slots. This option can be useful to limit the available hostkeys if the provider contains multiple keys and certificates.
- `no_poll(<yes|no>)` - by default the provider is polled once a second for changes. This option can be set if polling is not needed at all, for example if there is a static hostkey available from the provider.
- `polling_interval_ms(<time_ms>)` - defines the polling interval in milliseconds for the option above. Value range 100-86400000.
- `advertise(<yes|no>)` - by default the hostkey via PKCS#11 is not advertised to secure shell clients to aid with hostkey rotation. If this option is enabled, the external key will be advertised.
- `advertise_tectia_only(<yes|no>)` - if this option is enabled hostkey is advertised to Tectia clients only. Useful if some secure shell clients do not handle hostkey updates correctly.
- `allow-reuse(<yes|no>)` - if this option is enabled, the initialized PKCS#11 provider is reused upon configuration reload if the init string has not changed. Useful if the provider does not allow multiple initializations.
- `always-finalize(<yes|no>)` - if this option is enabled, multiple initizations of PKCS#11 provider are always finalized.
- `load-funcs-manually(<yes|no>)` - by default the function list is obtained from the provider. If this option is enabled, the function pointers are queried one by one instead.
- `always-login(<yes|no>)` - by default login is attempted when required by the provider. This option can be enabled for compatibility if provider omits login required.

For PKCS#12, the `<hostkey>` settings are as follows:

```
<hostkey>
  <externalkey type="software"
               init-info="key_file(/etc/ssh2/server-cert.p12)
                          key_passphrase_file(/etc/ssh2/my-passphrase)" />
</hostkey>
```

In the PKCS#12 sample output, the hostkey setting reads the PKCS#12 file `server-cert.p12` and if it needs a passphrase to open it, it will read the `my-passphrase` file and use the contents as the passphrase. The file can also contain additional certificates but they are ignored in Tectia Server.

In the `init-info` string, the following keywords are supported for the `software` type:

- `directory(<directory_name>)` - defines the directory to be polled for the keys. All files in the named directory are added to `sshexternalkey`. Note however, that this option lacks control over the actual server key and certificate.
- `polling_interval_ms(<time_ms>)` - defines the polling interval for the option above.
- `key_files(<key_spec>)` - defines that multiple comma-separated files are read. Loose grouping between files is expected so that public key, private key and certificate are assumed to be parts of the same key. Supported in Tectia Server.

- `key_file(<filename>)` - defines that one key file is read. The same as `key_files` with one parameter.
- `key_passphrase(<passphrase>)` - if a private key or certificate container is password-protected, the command tries to open it with the supplied passphrase first. In case the passphrase is not valid, the authentication callback is called normally. In the server, that means a failure to open the file as the server does not have an interactive prompt.
- `key_passphrase_file(<filename>)` - defines that instead of giving the passphrase in the configuration file directly, it can be written to a separate file. This option is useful if server configuration file needs to be more widely readable. The private key and passphrase can still be with root access only.

listener

This element is used to specify where the Secure Shell server listens for connections. The element has three attributes: `id`, `address`, and `port`.

The `id` must be given as an attribute. The value must be unique. The value must begin with a letter, it can contain alphanumeric characters or underscore characters but no whitespaces. Also the `port` and network interface `address` can be given. The default port for listeners is `22`.

Several listeners can be created to the same IP address to different ports. Each must have an unique ID. If the address is not specified, the server will listen to the given port on all interfaces.

Sample `listener` elements are shown below:

```
<listener id="internet" address="192.0.2.62" />
<listener id="intranet" address="10.0.0.1" />
<listener id="admin-private" port="222" />
```

Note

Ensure the firewall allows incoming connections to alternate port and any operating system restrictions, for example SELinux, do not prevent using the port. For example to allow alternate port '222' on SELinux system:

```
# semanage port --add --type ssh_port_t --proto tcp 222
```

domain-policy

On Windows, you can add this optional `domain-policy` element to define how Tectia Server will handle user names when a user logs on without specifying the prefix (indicating local or domain user). This element defines where the server will look for the user account, and how it will fill in the missing prefix part.

The `windows-domain-precedence` attribute defines a comma-separated list of trusted domains and special values `%default%` and `%local%`. The list is read in order, and the first domain that has an account for the user name will be used to log in the user and the rest will be ignored. If the user name is not found in any of the specified domains, the user account is assumed not to exist.

Value **%local%** means that a user without a specified prefix will be treated as a local user (`username` → `localmachine_name\username`).

Value **%default%** means that a user without a specified prefix will be treated as a domain user, and the domain name is expected to be the default domain of the local machine (`username` → `defaultdomain_name\username`).

If this element is not defined in the Tectia Server configuration, and a user logs on without specifying the prefix, Tectia Server first checks if the given user name is valid in the default domain where the local machine exists. If no match is found, for example because the machine is standalone, the user will be treated as a local user.

The `domain-policy` element can contain zero or more `windows-domain` elements.

windows-domain

This element defines domain user accounts for domain access with one-way trust. A one-way trust is a single, non-transitive trust relationship between two domains. In a one-way trust configuration between Tectia Server and a domain controller, the domain controller does not trust the Tectia Server process. The domain controller therefore refuses to give the server any information about the user that is trying to log on. Because the server does not know enough about the user, it refuses the logon procedure. You can use a domain user account to get this information from the domain controller.

For each `windows-domain` element, the `name` of the domain and `user` must be given as attributes. Set the password for the account either with the **`password-cache`** element or using the **Tectia Server Configuration** tool (for more information, see Section 4.1.5).

Note that you can only define one domain user account per domain.

The following example defines the `windows-domain-precedence` and a domain user account with one-way trust for user `sshadmin` in domain `SSH`:

```
<domain-policy
windows-domain-precedence="%local%, %default%, domainA, domainB">
 <windows-domain name="SSH" user="sshadmin" />
</domain-policy>
```

logging

This 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 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 setting a severity value for it.

For a complete list of log events, see Appendix D.

The following example sets the facility of the `Auth_method_failure` event to `auth` and the severity to `notice`. It also sets the facility of the `Server_reconfig_started` and `Server_starting` events to `discard` (the events will not be logged). All other events use the default settings.

```
<logging>
  <log-events facility="auth" severity="notice">
    Auth_method_failure
  </log-events>
  <log-events facility="discard">
    Server_reconfig_started Server_starting
  </log-events>
</logging>
```

limits

This element sets the maximum number of connections and processes the server will handle. Tectia Server uses a distributed architecture where the master server process launches several servant server processes that handle the actual connections. The element can also contain optional `servant-lifetime` element.

The `max-processes` attribute defines the maximum number of servant processes the master server will launch. The allowed value range is `1` to `2048`. The default (and recommended) value is `40`.

The `max-connections` attribute defines the maximum number of client connections (including `max-unauth-connections`) allowed per servant. The default (and recommended) value is `256`.

The `max-unauth-connections` and `max-unauth-connections-total` attributes define the maximum number of unauthenticated client connections allowed per servant and server's total unauthenticated client connections. The default value for both is `0`, that disables the unauthenticated connections limit.

The `max-new-connection-queue` attribute defines the maximum number of new client connections queued per servant before the server suspends accepting new connections. By default server does not limit passing new connections to the servant's queue and the value is set as `0`.

The maximum number of connections a server can handle depends on system resources.

This setting is useful in systems with low resources. The server has to be restarted to use the changed setting.

A sample `limits` element is shown below:

```
<limits max-unauth-connections="6" max-connections="256" max-processes="20" />
```

servant-lifetime

This element sets the total number of connections that a servant process will handle before the server process starts a new servant process in its place.

In some situations, the servant process may leak resources in OS provided services, resulting in resource starvation that prevents new connections. The `servant-lifetime` element will limit the maximum number of connections handled by the servant. After the maximum number of authenticated connections is handled, the server process starts retiring the servant. The server will not pass any new connections to the servant and launches a new servant instead, unless the maximum number of processes limit has been reached. The retired servant quits after its last existing connection disconnects. If the number of active running servants is below the preferred number of servants, a new servant process will be started.

You can check the status of the server and servant processes with the **ssh-server-ctl** tool and its **status** command. Use the verbose **-v** option for detailed connection listing per servant.

The `total-connections` attribute defines the total number of connections the servant process will handle during its lifetime. It can be any number from 1 to 4 billion. The recommended value is 5000.

If you do not give this option at all (default), the servants are never retired.

```
<limits max-connections="256" max-processes="5">
<servant-lifetime total-connections="5000"/>
</limits>
```

cert-validation

This element contains the CA certificates used in validation of the host-based and public-key authentication certificates. The element can have the following attributes: `http-proxy-url`, `socks-server-url`, `cache-size`, `max-crl-size`, `external-search-timeout`, `max-ldap-response-length`, `ldap-idle-timeout`, and `max-path-length`.

The `http-proxy-url` attribute defines a HTTP proxy and the `socks-server-url` attribute defines a SOCKS proxy for making LDAP or OCSP queries for certificate validity.

The address of the proxy 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 proxy) or `http://username@proxy_server:port/network/netmask,network/netmask ...` (with an HTTP proxy).

For example, by setting `socks-server-url` to *`"socks://mylogin@socks.ssh.com:1080/192.168.0.0/16,10.100.23.0/24"`*, the host *`socks.ssh.com`*

and port *1080* are used as your SOCKS server for connections outside of networks *192.168.0.0* (16-bit domain) and *10.100.23.0* (8-bit domain). Those networks are connected directly.

The `cache-size` attribute defines the maximum size (in megabytes) of in-memory cache for the certificates and CRLs. The allowed value range is 1 to 512, and the default value is 300 MB.

The `max-crl-size` attribute defines the maximum accepted size (in megabytes) of CRLs. Processing large CRLs can consume a considerable amount of memory and processing power, so in some environments it is advisable to limit their size. The allowed value range is 1 to 512, and the default value is 50 MB.

The `external-search-timeout` attribute defines the time limit (in seconds) for external HTTP and LDAP searches for CRLs and certificates. The allowed value range is 1 to 3600 seconds, and the default value is 60 seconds.

The `max-ldap-response-length` attribute defines the maximum accepted size (in megabytes) of LDAP responses. The allowed value range is 1 to 512, and the default value is 50 MB.

The `ldap-idle-timeout` attribute defines an idle timeout for LDAP connections. The validation engine retains LDAP connections and reuses them in forthcoming searches. The connection is closed only after the LDAP idle timeout has been reached. The allowed value range is 1 to 3600 seconds, and the default idle timeout is 30 seconds.

The `max-path-length` attribute limits the length of the certification paths when validating certificates. It can be used to safeguard the paths or to optimize against the paths getting too long in a deeply hierarchical PKI or when the PKI is heavily cross-certified with other PKIs. Using the attributes requires knowing the upper limit of the paths used in certificate validation. For example:

```
<cert-validation max-path-length="6">
  <ldap-server address="ldap://myldap.com" port="389" />
  <dod-pki enable="yes" />
  <ca-certificate name="CA 1" file="ca-certificate1.crt" />
</cert-validation>
```

In the example, the path is limited to six certificates, including the end-entity and root CA certificates. If not specified, the default value is 10. Decrease the value to optimize the validation if the maximum length of the encountered paths in the certificate validation is known.

The `cert-validation` element can contain **sub- elements**. You can use maximum one each of elements `cert-cache-file`, `crl-auto- update`, and `dod-pki`, and multiple instances of the other elements.

The validity of a received certificate is checked separately using each of the defined `ca-certificate` elements in turn until they are exhausted (in which case the authentication fails), or a positive result is achieved. If the certificate is valid, the `connections` and `authentication-methods` elements determine whether the certificate allows the user to log in (of course, the correct signature generated by a matching private key is always required in addition to everything else).

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 HTTP service address in URL format (`url`). Several OCSP responders can be specified by using several `ocsp-responder` elements.

For the OCSP validation to succeed, both the end-entity certificate and the OCSP responder certificate should be issued by the same CA. If different OCSP responder is used instead in the PKI environment, then the OCSP responder certificate itself or the CA certificate of its issuer must be configured in PEM format to enable trusted mode.

A sample `ocsp-responder` element for trusted mode is shown below:

```
<ocsp-responder
    validity-period="60"
    url="http://responder.example.com/ocsp/" >
    -----BEGIN TRUSTED MODE OCSP CA CERTIFICATE-----
    MIIDyjCCArKgAwIBAgIEBDH50zANBg...
    -----END CERTIFICATE-----
    </ocsp-responder>
```

If a certificate has an `Authority Info Access` extension with an OCSP Responder URL, it is only used if there are no configured OCSP responders. It is not used if any trusted mode OCSP responders have been configured, even if the certificate is unknown to the trusted mode OCSP responder.

The `validity-period` in seconds for the OCSP data can be optionally defined. During this time, new OCSP queries for the same certificate are not made but the old result is used.

If an OCSP 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.

cert-cache-file

This element specifies the name of the `file` where the certificates and CRLs are stored when the Tectia Server service is stopped, and read back in when the service is restarted.

On Unix, the cache file should be writable only by `root`.

On Windows, the cache file should be writable only by the **Administrators** group and the **SYSTEM** account.

crl-auto-update

This element turns on automatic updating of certificate revocation lists. When it is on, Tectia Server periodically tries to download the new CRL before the old one has expired. The `update-before` attribute can be used to specify how many seconds before the expiration the update takes place. The `minimum-interval` sets a limit for the maximum update frequency. The default minimum interval is `30` seconds.

crl-prefetch

This element instructs Tectia Server to periodically download a CRL from the specified `url`. The `url` 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, 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

One of the compliance requirements of the US Department of Defense Public-Key Infrastructure (DoD PKI) is to have the Digital Signature bit set in the Key Usage of the certificate. To enforce digital signature in key usage, set the value of the `enable` attribute to `yes`. The default is `no`.

ca-certificate

This element enables user authentication using certificates. It can have five attributes: `name`, `file`, `disable-crls`, `use-expired-crls`, and `trusted`.

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 element.

CRL checking can be disabled by setting the `disable-crls` attribute to `yes`. The default is `no`.

Note

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

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).

The CA certificate is by default set as a trust anchor and it is trusted explicitly (`trusted="yes"`). No revocation checks are performed on the CA certificate (only the validity period will be checked), and it will be the end point of the validation path, meaning that no CA above it in the PKI hierarchy will affect the validation. If the `trusted` attribute is set to `no`, the CA will be considered an intermediate CA. At least one trusted CA certificate is required for a working PKI setting.

openssh-ca-key

This element enables user authentication using OpenSSH CA-key. It can have two attributes: `name`, and `file`.

The `name` must contain the name of the CA.

The element must either contain the path to the OpenSSH CA-key file as a value of the `file` attribute, or include the certificate as a base64-encoded ASCII element.

Generic cert-validation elements do not apply to OpenSSH certificate validation, as there is no revocation checking.

A sample `cert-validation` element is shown below:

```
<cert-validation   http-proxy-url="http://proxy.example.com:800">
  <ldap-server     address="ldap.example.com"
                   port="389" />
  <ocsp-responder  validity-period="60"
                   url="http://ca.example.com/ocsp-1/" />
  <cert-cache-file file="/var/cert-cache.dat" />
  <crl-auto-update update-before="30"
                   minimum-interval="600" />
  <crl-prefetch    interval="1800"
                   url="http://ca.example.com/default.crl" />
  <dod-pki         enable="no" />
  <ca-certificate  name="exa-ca1"
                   file="/etc/ssh2/exa-ca1.crt" />
  <ca-certificate  name="exa-ca2"
                   file="/etc/ssh2/exa-ca2.crt"
                   use-expired-crls="3600" />
  <ca-certificate  name="testonly-ca"
                   file="/etc/ssh2/testonly-ca.crt"
                   disable-crls="yes" />
  <openssh-ca-key  name="exa-openssh-ca1"
                   file="/etc/ssh2/exa-openssh-ca1.pub" />
</cert-validation>
```

password-cache

On Windows, this element specifies the location of the server password cache file, given as the value of the `file` attribute.

```
<password-cache file="C:\Program Files (x86)\SSH Communications Security\SSH Tectia\
                     SSH Tectia Server\sshpwcache.db"/>
```

For more information, see Section 4.1.5.

load-control

The `load-control` element defines settings for keeping Tectia Server working when the load is high, that is, the number of current connections is near `max-connections` (the maximum number of client connections allowed per servant, specified in the `limits` element). High load might be caused by a connection flood denial-of-service attack that tries to make the server unavailable to its intended users by using so much of its resources that normal service is disrupted.

Load control is implemented by keeping a "white list" of the IP addresses of connections that have had a successful authentication. When Tectia Server starts, the white list is empty. When the server's load is high, connections from IP addresses that are not on the white list (that is, connections that have not recently had a successful authentication) are discarded.

The `load-control` element can have three attributes: `enable`, `discard-limit`, and `white-list-size`.

The `enable` attribute can have a value of `yes` or `no`. The default value is `yes` (load control is enabled).

Note

If the maximum number of client connections allowed per servant is set to 1 (`max-connections="1"`), load control is disabled even if you have set `load-control enable="yes"` in the configuration file.

When the number of a servant's concurrent connections is not higher than the value of `discard-limit`, the servant accepts connections from any IP address. When the number of a servant's concurrent connections exceeds the value of `discard-limit`, only connections from IP addresses that are on the server's white list are accepted. If existing servants cannot accept any more connections, but the maximum number of servant processes the master server will launch (specified with the `max-processes` attribute of the `limits` element) has not been reached, the server launches a new servant which will accept new connections.

The allowed value range for `discard-limit` is 1 to `max-connections`-1. The default value of `discard-limit` depends on the value of `max-connections`. The default values of `discard-limit` for different values of `max-connections` are the following:

max-connections	**discard-limit default value**
1	N/A (`load-control` is disabled)
2-10	`max-connections` - 1
>10	0.9 * `max-connections`

If you have not defined any Tectia Server configuration settings (that is, only default values are used), the value of `max-connections` is 256 and the value of `discard-limit` is 230 (that is, 90 percent of `max-connections`).

The `white-list-size` attribute specifies the number of IP addresses on the server's white list. The allowed value range for `white-list-size` is 1 to 10000. The default value is 1000.

The `connections` Block

The `connections` block defines the basic rules for allowing and denying connections. The `connections` block includes one or more `connection` elements.

If a user does not match to any selectors in the `connection` elements, the connection is allowed with server default connection settings.

connection

Each `connection` element specifies either an allow or deny rule for connections. The element can have three attributes: `name`, `action`, and `tcp-keepalive`.

The word `allow` or `deny` is given as a value of the `action` attribute. By default, if the `action` attribute is omitted, the connection is allowed.

The `name` attribute can be used to give an identifier to the connection rule. The value must be a valid XML name beginning with a letter and containing alphanumeric characters or the underscore character without any whitespace. The identifier can be used, for example, in auditing.

The `tcp-keepalive` attribute defines whether the system should send keepalive messages to the other side. If they are sent, a broken connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily, and this can be annoying in some situations. On the other hand, if keepalive messages are not sent, sessions may hang indefinitely on the server, leaving "ghost" users and consuming server resources. The value must be `yes` or `no`. The default is `no` (do not send keepalives).

The `connection` element can include one or more selectors, a rekey setting, and one or more cipher, MAC, KEX and host key algorithm definitions.

selector

The selectors define to which connections this connection rule applies to. Only the `interface` and `ip` selector attributes can be used in the `connections` block. Other information, for example the user name, is not yet available at this stage of the connection. Do not define any other selector attributes, as their matching process would end in error and terminate the connection attempt. See Section 4.2.2.

interface

This selector matches to the listener interface `id` or `address` and/or `port`. At least one attribute must be given. If the `id` is defined, the others MUST NOT be given. If the `id` is not defined, either or both of `address` and `port` may be given.

ip

This selector matches to an IP address or FQDN (fully qualified domain name) of the client. Either `address` or `fqdn` can be given, not both.

The `address` can be in one of the following formats:

- 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 `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns. These patterns may also contain "*" and "?" globbing characters.

rekey

This element specifies the number of `seconds` or transferred `bytes` after which the key exchange is done again.

If a value for both `seconds` and `bytes` is specified, rekeying is done whenever one of the values is reached, after which the counters are reset.

The defaults are `3600` seconds (1 hour) and `1000000000` bytes (~1 GB). The value `0` (zero) turns rekey requests off. This does not prevent the client from requesting rekeys.

cipher

This element selects a cipher `name` allowed by the server for data encryption.

The list of supported ciphers can be found in Section G.1.

Multiple ciphers can be specified by using multiple `cipher` elements.

Normally when a specified cipher is not found on the server, the configuration file reading fails and the server will not restart. The `cipher` element may optionally take an `allow-missing` attribute, which can have a value of `yes` or `no`. If a value of `yes` is given for this attribute and a specified cipher is not found during configuration reading (for example, CryptiCore on Solaris), the server logs a warning to the syslog but will restart normally. The default is `no` (a missing cipher is treated as fatal error and the server configuration reading fails).

Setting the `allow-missing` attribute to `yes` is useful when you want to use the same `ssh-server-config.xml` file on multiple servers and only some of the servers have, for example, CryptiCore available.

mac

This element selects a MAC `name` allowed by the server for data integrity verification.

The list of supported MACs can be found in Section G.3.

Multiple MACs can be specified by using multiple `mac` elements.

Normally when a specified MAC is not found on the server, the configuration file reading fails and the server will not restart. The `mac` element may optionally take an `allow-missing` attribute, which can have a value of `yes` or `no`. If a value of `yes` is given for this attribute and a specified MAC is not found during configuration reading (for example, CryptiCore on Solaris), the server logs a warning to the syslog but will restart normally. The default is `no` (a missing MAC is treated as fatal error and the server configuration reading fails).

Setting the `allow-missing` attribute to `yes` is useful when you want to use the same `ssh-server-config.xml` file on multiple servers and only some of the servers have, for example, CryptiCore available.

kex

This element selects a KEX `name` allowed by the server for key exchange method.

The list of supported classical and PQC hybrid KEXs can be found in Section G.2.

Multiple KEXs can be specified by using multiple `kex` elements.

hostkey-algorithm

This element selects a host key signature algorithm `name` to be used in server authentication with host keys or certificates.

Multiple host key algorithms can be specified by using multiple `hostkey-algorithm` elements. The host key algorithms are tried in the order they are specified.

The algorithms to be used are configured in both the Connection Broker and Tectia Server configuration files. The algorithms that will be used are those that the server offers in negotiation based on the available host keys and/or certificates and the first common algorithm that the client prefers. This way the use of only certain algorithms, such as SHA-2, can be enforced by the server.

The list of supported host key signature algorithms can be found in Section G.4.

A sample `connection` element that allows connections from a specified IP address range is shown below:

```
<connection name="conn1" action="allow" tcp-keepalive="yes">
  <selector>
    <ip address="192.168.0.42-192.168.0.82" />
  </selector>
  <rekey seconds="600" bytes="500000000" />
  <cipher name="crypticore128@ssh.com" />
  <mac name="crypticore-mac@ssh.com" />
  <kex name="diffie-hellman-group-exchange-sha256" />
  <hostkey-algorithm name="ssh-dss-sha256@ssh.com" />
</connection>
```

A sample `connection` element that denies all connections is shown below. As the element does not contain any selectors, it matches always. This can be used as the last element in the `connections`

element to deny all connections that were not explicitly allowed by the previous elements. (By default, non-matching connections would be allowed.)

```
<connection action="deny" />
```

The `authentication-methods` Block

The `authentication-methods` block defines the authentication methods that are allowed and required by the server. It can have one attribute: `login-grace-time`. It can contain a `banner-message` and an `auth-file-modes` element and multiple `authentication` elements.

The `login-grace-time` attribute is used to specify a time after which the server disconnects if the user has not successfully logged in. If the value is set to `0`, there is no time limit. The default is `600` (seconds).

The authentication methods that are on the same level under one `authentication` element are considered allowed (one of them must succeed).

Several authentication methods can be set as required by nesting the `authentication` elements inside each other.

The server allows by default public-key, keyboard-interactive, and password authentication (one of them must succeed).

If the `authentication-methods` element is empty or missing from the `ssh-server-config.xml` file, the server does not allow any authentication methods and the configuration is essentially defunctional.

However, if you put inside `authentication-methods` an `authentication` element with no authentication methods defined, the matching users will be allowed to log in without authentication. This can be used in combination with selectors and/or nested authentication methods, but should never be used as the only authentication element.

Caution

Consider carefully before putting an empty `authentication` element in the `ssh-server-config.xml` file. It will allow the matching users (everyone, if no selectors are used) to log in without authentication.

banner-message

This element specifies the path to the message that is sent to the client before authentication. The path is given as a value of the `file` attribute. Alternatively, the banner message can be given as the contents of the `banner-message` element.

Note, however, that the client is not obliged to show this message or any banner message sent by the Tectia Server. See also **blackboard** for sending a banner message later during the authentication process.

```
<banner-message file="/etc/ssh2/banner-message">
  This is the server banner message. If file attribute is set,
  this inlined text is ignored, and the file is read instead
```

```
  (like in this example).
</banner-message>
```

auth-file-modes

This element specifies whether Tectia Server on Unix platforms should check permissions and ownership of the user's key files used for public-key authentication or the directory where they are stored.

The word `yes` or `no` is given as a value of the `strict` attribute. If set to `yes`, the permissions and ownership of the `.ssh2` directory, the `.ssh2/authorization` file (if used), the `.ssh2/authorized_keys` directory (if used), and the keys listed in the `authorization` file or present in the `authorized_keys` directory are checked.

This is normally desirable because users sometimes accidentally leave their directory or files world-writable, in which case anyone can edit the authorization and key files. The default is `yes`.

The `mask-bits` attribute can be used to specify the forbidden permission bits in octal format. This setting controls both the file and directory permissions when used without `dir-mask-bits`. The default is `022` (group and others must not have the write permission).

The ownership of the checked files and directories must be either root or the user.

The value of `mask-bits` is given with 3 digits:

```
<auth-file-modes strict="yes" mask-bits="022" />
```

The `dir-mask-bits` attribute can be used to specify the forbidden permission bits in octal format (with 4 digits) for the directory where the user's key files are stored.

If only `dir-mask-bits` is defined, the value of `mask-bits` is assumed to be 022, and it is only applied to files.

```
<auth-file-modes strict="yes" dir-mask-bits="0222"/>
```

When the server has been configured to use strict mode for public-key authentication, you can define different permissions for the user's key files and for their directory. In this case, define both the `mask-bits` and `dir-mask-bits` settings together, for example as follows:

```
<auth-file-modes strict="yes"
                 mask-bits="222"
                 dir-mask-bits="0222"/>
```

authentication

Each `authentication` element specifies a chain of authentication methods. It can include one or more selectors and different authentication methods. It may also include other `authentication` elements.

Defining nesting `authentication` elements within each other sets the child elements as *required* (all must be passed for the authentication to be successful). Setting multiple authentication methods

at the same level sets them as *optional* (one of the methods must be passed for the authentication to be successful).

The `authentication` elements are read in top-down order. For elements on the same level, the first matching element is used and the remaining elements are ignored. If the element has nested child elements, they are matched next using the same procedure.

In the `authentication` element, the `action` attribute takes values `allow` or `deny`. The `allow` value means that users who match a selector will be allowed into the system. The `deny` value means that access will be denied from users who match a selector. By default, if the `action` attribute is omitted, authentication is allowed.

If an authentication chain ends in a `deny` action, or if the user does not match to any selectors in the `authentication` elements, the user is not allowed to log in.

Note

Note that the behavior has changed in Tectia Server 5.1. In Tectia Server 5.0, a non-matching user was allowed the default authentication methods.

In a nested chain of `authentication` elements, it is possible, for example, to set the parent method to deny authentication and a child element with a selector to allow authentication. If the user matches the selector and successfully completes the authentication method(s), login is allowed.

For more information on using authentication chains, see Section 5.12.

The `authentication` element can additionally take a `set-group` attribute, which sets a group for the users that pass the particular authentication chain. The group definition can be later used in the `services` element.

If `set-group` is used here, it overrides any `group` definitions in the `services` element. See the section called “The `services` Block”.

On Windows, the `authentication` element can take a `password-cache` attribute with values `yes` or `no`. This can be used to enable or disable password caching for the authentication block. By default, password caching is disabled (set to `no`). For more information, see Section 4.1.5.

The authentication `name` can be optionally given as an attribute. The value of `name` must be a valid XML name beginning with a letter and containing alphanumeric characters or the underscore character without any whitespace. The authentication `name` can be used, for example, in auditing.

Caution

Consider carefully before putting an empty `authentication` element in the `ssh-server-config.xml` file. It will allow the matching users (everyone, if no selectors are used) to log in without authentication.

selector

The selectors define to which connections this authentication method applies to. All selectors can be used in the `authentication-methods` block. See Section 4.2.2.

certificate

This selector defines the information that needs to be matched in the specified `field` of user certificates used in public-key authentication. The information to match is specified in attribute `pattern` or `regexp` (regular expression). Do not define both in the same selector!

Using this selector requires that the parent element in the authentication chain contains an `auth-publickey` element.

The field can be either `ca-list`, `issuer-name`, `subject-name`, `serial-number`, `altname-email`, `altname-upn`, `altname-ip`, `altname-fqdn`, or `extended-key-usage`.

The format of the pattern depends on the type of the field. The `ca-list` field contains a list of CA names separated by commas. The names that are defined in the `ca-certificate` element are used. The `issuer-name` and `subject-name` fields contain distinguished names, `serial-number` a positive integer. The `altname-fqdn` field contains a host name and `altname-ip` an IP address or a range. The `altname-email` field contains an email address and `altname-upn` the principal name.

The `extended-key-usage` setting can be used to define the allowed key purposes for certificates. The main purpose of this option is to prevent authentication with wrong certificate types, for example a user certificate should not be accepted for host-based authentication. In the `extended-key-usage` field, add a comma-separated list of standard names or numerical OIDs that specify which certificate key purposes will be accepted:

```
Standard name        OID                      Description
-------------------------------------------------------------------------
serverAuth           1.3.6.1.5.5.7.3.1        TLS WWW server authentication
clientAuth           1.3.6.1.5.5.7.3.2        TLS WWW client authentication
ssh-server           1.3.6.1.4.1.2213.15.1.1  SecSH server authentication
                                              (host certificate)
ssh-client           1.3.6.1.4.1.2213.15.1.2  SecSH user authentication
ssh-clientHostbased  1.3.6.1.4.1.2213.15.1.3  SecSH hostbased authentication
```

For a configuration example, see the section called “Authentication Examples”.

With `extended-key-usage`, the `explicit` attribute can be used to request that the certificate must include the key purpose ID specified with the pattern. If this attribute is not used, any certificate containing no key purpose ID or containing the `anyExtendedKeyUsage` definition will be accepted. For information on `anyExtendedKeyUsage` and its usage, see *RFC3280, Section 4.2.1.13: Extended Key Usage*.

The `altname-fqdn`, `altname-upn`, `altname-email`, `subject-name`, and `issuer-name` selectors may contain the `%username%` keyword which is replaced with the user's login name before comparing with the actual certificate data. On Windows, the `%username-without-domain%` keyword can be used and it is replaced by the user's login name without the domain

part. The `%hostname%` keyword can be used in the same way and it is replaced by the client's FQDN. These patterns may also contain "*" and "?" globbing characters.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the `pattern-case-sensitive` attribute.

In the `regexp` attribute, you can define a regular expression to match a range of values in the selected `field`. Regular expressions follow the egrep syntax.

For the `issuer-name` and `subject-name` selectors, you can also define if the pattern has to match the subject name completely or only partly. Use the `ignore-prefix` attribute to match only the end of subject name, and the `ignore-suffix` attribute to match only the beginning of the subject name. The ignore options are optional.

You can also define both of the ignore options on simultaneously in which case the pattern has to match with some point in the subject name. For example: when both ignore settings are defined on, pattern `O=SSH,OU=*,CN=example` matches with:

```
C=FI, O=SSH, OU=RandD, CN=example, CN=UID12345
```

The `allow-undefined` attribute can be used to control the behavior of the selector when the required certificate data is not defined (certificates have not been used at all, or the certificate does not contain the fields to be matched). Its value must be `yes` or `no`. If set to `yes`, the undefined data is treated as non-matched and the matching continues to other elements. The default is `no`, (trying to match undefined data results in termination of the connection). For more information, see the section called “Selectors and Undefined Data”.

Caution

When creating the certificate selectors, make sure that every `selector` element ties the user name to the certificate, either by including a `user` sub-element, or by putting the special substitution string `%username%` or `%username-without-domain%` to a field used to match the correcponding field in the certificate.

Failing to do this may cause unintended consequences, for example authentication succeeding with many different user names with a single certificate.

host-certificate

This selector defines the information that needs to be matched in the specified `field` of host certificates used in public-key authentication. The information to match is specified in attribute `pattern` or `regexp` (regular expression). Do not define both in the same selector!

Using this selector requires that the parent element in the authentication chain contains an `auth-hostbased` element.

The field can be either `ca-list`, `issuer-name`, `subject-name`, `serial-number`, `altname-email`, `altname-upn`, `altname-ip`, `altname-fqdn`, or `extended-key-usage`.

See details of the field contents under the `certificate` selector.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the `pattern-case-sensitive` attribute.

In the `regexp` attribute, you can define a regular expression to match a range of values in the selected `field`. Regular expressions follow the egrep syntax.

For the `issuer-name` and `subject-name` selectors, you can also define if the pattern has to match the subject name completely or only partly. Use the `ignore-prefix` attribute to match only the end of the subject name, and the `ignore-suffix` attribute to match only the beginning of the subject name. Both attributes can be used together in which case the pattern has to match with some point in the subject name. The ignore options are optional.

The `allow-undefined` attribute can be used to control the behavior of the selector when the required certificate data is not defined (certificates have not been used at all, or the certificate does not contain the fields to be matched). Its value must be `yes` or `no`. If set to `yes`, the undefined data is treated as non-matched and the matching continues to other elements. The default is `no`, (trying to match undefined data results in termination of the connection). For more information, see the section called “Selectors and Undefined Data”.

interface

This selector matches to the listener interface `id` or `address` and/or `port`. At least one attribute must be given. If the `id` is defined, the others MUST NOT be given. If the `id` is not defined, either or both of `address` and `port` may be given.

ip

This selector matches to an IP address or FQDN (fully qualified domain name) of the client. Define the information to be matched with either the `address` or the `fqdn` or the `fqdn-regexp` attribute, but do not use them together.

The `address` can be in one of the following formats:

- 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 `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns.

In the `fqdn-regexp` attribute, you can define a regular expression to match a range of FQDNs. Regular expressions follow the egrep syntax.

user

This selector matches to a user name or ID. Define the information to be matched with one of the following attributes (do not use them together): `name`, `id` or `name-regexp`.

In attribute `name`, you can define a comma-separated list of user names.

In attribute `id`, you can define a comma-separated list of user IDs.

In attribute `name-regexp`, you can define a regular expression to match a range of names. Regular expressions follow the egrep syntax.

Names are normally matched case-insensitively. Alternatively, you can define the names to be taken exactly as entered by using the `name-case-sensitive` attribute.

In Windows domain environment, the `user` and `user-group` selectors have a length limitation. For more information, see the description of option **User** in Section 4.1.

user-group

This selector matches to a user group name or ID. Define the information to be matched with either the `name` or the `id` or the `name-regexp` attribute, but do not use them together.

In attribute `name`, you can define a comma-separated list of user names.

In attribute `id`, you can define a comma-separated list of user IDs.

In attribute `name-regexp`, you can define a regular expression to match a range of user group names. Regular expressions follow the egrep syntax.

Names are normally matched case-insensitively. Alternatively, you can define the names to be taken exactly as entered by using the `name-case-sensitive` attribute.

In Windows domain environment, the `user` and `user-group` selectors have a length limitation. For more information, see the description of option **User** in Section 4.1.

user-privileged

This selector matches to a privileged user (administrator or root) or to a non-privileged user. The `value` can be `yes` (match to a privileged user) or `no` (match to a normal user).

The `allow-undefined` attribute can be used to control the behavior of the selector when the required data is not defined (user-privilege level is not known). Its value must be `yes` or `no`. If set to `yes`, the undefined data is treated as non-matched and the matching continues to other elements. The default is `no`, (trying to match undefined data results in termination of the connection). For more information, see the section called “Selectors and Undefined Data”.

Note

On a Windows server, the user-privilege level is not available during the authentication phase when the user is logging in using a domain account and does not yet have an access token allocated. To get the user-privilege status for domain users, the user should first pass password or GSSAPI authentication.

If the privilege level needs to be checked for local accounts, the `allow-undefined` attribute should be set to `yes` or else connection fails for users logging in using domain accounts. However, this means that the user-privilege status will not be verified for Windows domain users.

To check the privilege level of domain accounts on a Windows server in the `authentication-methods` block, the `user-privileged` selector should be used in a nested authentication block when password or GSSAPI authentication has already been passed.

blackboard

This selector matches to information in the specified blackboard `field`. The information to match can be defined with attribute `pattern` or `regexp`. Do not define both in the same selector!

During the connection setup phase, the server stores information on the various parameters of the client (for example, IP address, user name, certificate fields), which can be later used to allow/deny the connection and set the connection parameters. The information is stored in the so called *blackboard fields*. The blackboard fields are typically used as elements inside the *selectors*.

The most commonly used selector attributes have their own sub-elements. Custom attributes can be specified with the generic `blackboard` sub-element.

Patterns are normally matched case-insensitively. Alternatively, the pattern can be specified using the `pattern-case-sensitive` attribute.

In the `regexp` attribute, you can define a regular expression to match a range of values in the selected `field`. Regular expressions follow the egrep syntax.

The `allow-undefined` attribute can be used to control the behavior of the selector when the required data is not defined. Its value must be `yes` or `no`. If set to `yes`, the undefined data is treated as non-matched and the matching continues to other elements. The default is `no`, (trying to match undefined data results in termination of the connection). For more information, see the section called “Selectors and Undefined Data”.

Blackboard also supports `banner-message` for sending additional messages at certain stage(s) of the authentication process. You may use substitutions to include other blackboard values in the message. The banner message also supports the `$bbdump` substitution for displaying all blackboard data. For descriptions of various blackboard fields, please see more details in **mapper** element.

Example blackboard selector and banner-message configuration:

```
<authentication action="allow" name="authentication-chain">
 <auth-publickey/>
 <authentication name="authenticated_user_banner_message">
  <selector>
   <blackboard field="protocol-kex"
               pattern="mlkem1024nistp384-sha384"/>
   <blackboard field="protocol-kex"
               pattern="mlkem768nistp256-sha256"/>
   <blackboard field="protocol-kex"
```

```
              pattern="mlkem768x25519-sha256"/>
  <blackboard field="protocol-kex"
              pattern="ecdh-nistp521-kyber1024-sha512@ssh.com"/>
  <blackboard field="protocol-kex"
              pattern="curve25519-frodokem1344-sha512@ssh.com"/>
  <blackboard field="protocol-kex"
              pattern="sntrup761x25519-sha512@openssh.com"/>
  <blackboard field="protocol-kex"
              pattern="curve448-kyber1024-sha512@ssh.com"/>
 </selector>
 <set-blackboard field="banner-message">
       Welcome dear %username%!
       Some info about your new quantum-safe (as of $time) session:
        KEX=$protocol-kex
        HostkeyAlg=$protocol-hostkey-algorithm
        Cipher=$protocol-cipher-c2s/$protocol-cipher-s2c
        ClientVersion=$peer-version
        SessionId=$session-id

        Your publickey info:
         $publickey-type $publickey-length

        Your X.509v3 certificate info:
         certificate-issuer-name=$certificate-issuer-name
         certificate-subject-name=$certificate-subject-name
         certificate-serial-number=$certificate-serial-number

        ServerVersion=$product-version
         $product-name [%product-os%]

        BLACKBOARD DUMP FOR TESTING:
        $bbdump
 </set-blackboard>
</authentication>
</authentication>
```

publickey-passed

This selector matches if authentication is passed using a normal public key (without a certificate). Using this selector requires that the authentication chain contains an `auth-publickey` element.

Optionally, the `length` range of the public key can be given as an attribute, for example `"2048-4096"` (keys from 2048 to 4096 bits match). The range can also be left open, for example `"3072-"` (keys over 3072 bits match).

The `allow-undefined` attribute can be used to control the behavior of the selector when the required data is not defined (public-key authentication has not been used). Its value must be `yes` or `no`. If set to `yes`, the undefined data is treated as non-matched and the matching continues to other elements. The default is `no`, (trying to match undefined data results in termination of the connection). For more information, see the section called “Selectors and Undefined Data”.

user-password-change-needed

On Unix platforms, this selector matches if the user password has expired and should be changed. For more information, see the section called "Forcing Password Change".

The `allow-undefined` attribute can be used to control the behavior of the selector when the required data is not defined. Its value must be `yes` or `no`. If set to `yes`, the undefined data is treated as non-matched and the matching continues to other elements. The default is `no`, (trying to match undefined data results in termination of the connection). For more information, see the section called "Selectors and Undefined Data".

set-blackboard

The `set-blackboard` element can be used to describe an item that will be added to the blackboard immediately when this authentication block is encountered. Even if the block does not complete the authentication, the added fields will persist in the blackboard.

The `set-blackboard` element takes a mandatory attribute `field` which gives the blackboard key where the item is stored. Any previous data in the location will be overridden. The attribute value can be given with the mutually exclusive options:

`value` which defines the desired value

`file` which defines a path to a file containing the desired value

`<PCDATA>` directly in the element.

set-user

The `set-user` element can be used to define that a specified user name will be used from here on. Selectors (`user`, `user-group`, etc.) will use value specified with this element. The `set-user` element can occur multiple times in an authentication chain (but only once in one `authentication` block).

The changed user name is defined in the `name` attribute.

The value specified here will be persistent, and will take effect immediately after any methods specified in the block have been successfully completed. Any enclosed `authentication` blocks will use the new value.

The connection will be disconnected if the specified user account does not exist.

auth-publickey

This element sets the public-key authentication method. The element can have the following attributes: `require-dns-match`, `signature-algorithms`, `authorization-file`, `authorized-keys-directory`, and `openssh-authorized-keys-file`.

The `require-dns-match` attribute is used to accept or deny a public key which has the `allow/deny-from` option set in the authorization file. If the attribute is set to `yes`, an additional check

for a properly configured DNS is made at the moment when the `allow/deny-from` option is processed. That is, the host name lookup must succeed for the connection to be accepted. If the attribute is set to `no` (default), the DNS host name for the client's IP address is ignored. This attribute corresponds to `RequireReverseMapping` and is for compatibility with SSH Tectia Server versions 4.x.

A configuration example:

```
<auth-publickey require-dns-match="yes" />
```

Note

A failure will always result in case of the following configuration settings: `resolve-client-hostname="no"` and `require-dns-match="yes"`.

The `signature-algorithms` attribute can be used to specify a comma-separated list of public-key signature algorithms used for user authentication. The algorithms that can be used are those that are defined in both Tectia Server and Connection Broker configuration files. They are defined in an order of preference. The one that is selected to be used, is the first common algorithm that both the client and server have in their configuration. This way the use of only certain algorithms, such as SHA-2, can be enforced. The supported and default algorithms are the same as those listed for **hostkey-algorithm** .

A public-key signature algorithm configuration example:

```
<auth-publickey
    signature-algorithms="ssh-rsa,ssh-dss-sha256@ssh.com" />
```

The `authorization-file` attribute can be used to specify a comma-separated list of paths to files that contain the user public keys that are authorized for login. The paths can contain pattern strings that are expanded by Tectia Server.

The following pattern strings can be used:

- `%D` or `%homedir%` is the user's home directory
- `%U` or `%username%` is the user's login name

  For Windows domain users, these strings are substituted differently:

  - `%U` is expanded to `domain.username`
  - `%username%` is expanded to `domain\username`
- `%IU` or `%userid%` is the user's user ID (uid)
- `%IG` or `%groupid%` is the user's group ID (gid)

Note that user ID and group ID are only supported on Unix, not on Windows.

The default is `%D/.ssh2/authorization`.

The `authorized-keys-directory` attribute can be used to specify a comma-separated list of directories that contain the user public keys that are authorized for login. As above, the paths can contain pattern strings that are expanded by Tectia Server. The default is `%D/.ssh2/authorized_keys`.

The `openssh-authorized-keys-file` attribute can be used to specify a comma-separated list of paths to OpenSSH-style `authorized_keys` files that contain the user public keys that are authorized for login. As above, the paths can contain pattern strings that are expanded by Tectia Server.

Tectia Server looks for a matching public-key in the following locations in the following order:

1. In the defined authorization file, if such a file exists.
2. In the defined `authorized_keys` directory, if no authorization file is available or the defined file does not include a matching public-key.
3. In the defined OpenSSH authorized-keys file, if no matching key was found in the Tectia -related authorization file or key directory.
4. In the `authorization` file and then in the `authorized_keys` directory located in the directory defined in `user-config-dir` in the `settings` element, if none of the above locations produced a matching key.
5. In the default public-key storage location, if none of the previous settings was made.

For more information, see Section 5.6.

auth-hostbased

This element sets the host-based authentication method. The element can take attributes: `require-dns-match`, `disable-authorization` or `allow-missing`.

If the `require-dns-match` attribute is set to `yes`, host-based authentication will require the host name given by the client to match the one found in DNS. If the host name does not match, the authentication will fail. The default is `no` (exact match not required).

If the `disable-authorization` attribute is set to `yes`, host-based authentication ignores authorization requirements. This is applicable for troubleshooting and testing purposes The default is `no` (authorization is enabled).

Normally the `auth-hostbased` is required in the server configuration. The `allow-missing` attribute may optionally be used, and when set to `yes` the server ignores the missing element. The default is `no` (a missing `auth-hostbased` is treated as a fatal error and the server configuration reading fails).

auth-password

This element sets the password authentication method.

The delay between failed attempts in seconds (`failure-delay`) and the maximum number of attempts (`max-tries`) can be given as attributes. The default delay is 2 seconds and default maximum is 3 attempts.

auth-keyboard-interactive

This element sets the keyboard-interactive authentication method.

The delay between failed attempts in seconds (`failure-delay`) and the maximum number attempts (`max-tries`) can be given as attributes. The default delay is 2 seconds and default maximum is 3 attempts.

The keyboard-interactive submethods are given as child elements. The supported methods are `submethod-password`, `submethod-pam`, `submethod-securid`, `submethod-radius`, `submethod-aix-lam`, and `submethod-generic`.

If no submethods are configured, all available submethods are allowed by default (however, the server may not be able to find the necessary libraries for SecurID and PAM, for example). If some of the submethods are configured, the rest of the submethods are implicitly disabled.

submethod-pam

This element sets the keyboard-interactive PAM submethod in use. PAM is supported on Unix platforms.

The `service-name` attribute can be used to instructs PAM about which configuration it should use. When used, this setting will override the setting in `pluggable-authentication-modules`. Note that it is possible to define different service names for authentication and user session management by defining different values for the `service-name` here and in the `pluggable-authentication-modules` element.

If you have multiple `authentication` elements that have the PAM submethod enabled (for example for different authentication chains for intranet and Internet users), you can define different `service-name` settings for them.

The `dll-path` can be used to define a non-standard location for the PAM library, or a comma-separated list of PAM DLLs. If the PAM library is not in the default library path then the `dll-path` attribute is needed both here in `submethod-pam` and in the `pluggable-authentication-modules` element.

On AIX, the path should include the archive file, unless the library is a shared object or has been extracted from the shared object.

submethod-password

This element sets the keyboard-interactive password submethod in use.

submethod-securid

This element sets the keyboard-interactive SecurID submethod in use.

The `dll-path` to the SecurID DLL can be given in an attribute. The path must point to the operating-system specific SecurID module, for example, `"/usr/lib/libaceclnt.so"` on Solaris.

submethod-radius

This element sets the keyboard-interactive RADIUS submethod in use.

The element can contain multiple `radius-server` child elements.

radius-server

This element defines a RADIUS server. The element has four attributes: `address`, `port`, `timeout`, and `client-nas-identifier`.

The `address` is the IP address of the RADIUS server. The `port` is the RADIUS server port. The default port is `1812`.

The `timeout` is the time in seconds after which the RADIUS query is terminated if no response is gained. The default is `10` seconds.

The `client-nas-identifier` attribute defines the network access server (NAS) identifier to be used when talking to the RADIUS server.

The element must contain one `radius-shared-secret` child element.

radius-shared-secret

This element defines the RADIUS shared secret file.

The path to the secret file can be given as a value of the `file` attribute.

Alternatively, the secret can be included as the contents of the `radius-shared-secret` element.

submethod-aix-lam

This element enables Tectia Server to use LAM directly on AIX platforms. LAM is used as a keyboard-interactive plugin. By default, the LAM authentication submethod is enabled.

The **submethod-aix-lam** takes an optional attribute `enable-password-change` with value `yes` or `no`. By default password changes are not enabled. To enable LAM on AIX, and to allow users to change their expired passwords, use the following settings:

```
<authentication-methods>
  <authentication name="authentication">
    <auth-keyboard-interactive >
      <submethod-aix-lam
          enable-password-change="yes" />
    </auth-keyboard-interactive >
  </authentication>
</authentication-methods>
```

submethod-generic

This element sets the generic submethod in use. This element can be used to add custom submethods to keyboard-interactive authentication.

The `name` on the method must be given in the attribute.

Optional `params` for the submethod can be given as well.

auth-gssapi

This element sets the GSSAPI authentication method.

The `dll-path` can be given as an attribute. This specifies where the necessary GSSAPI libraries are located. If this attribute is not specified, the libraries are searched for in a number of common locations. The full path to the libraries should be given, for example, `"/usr/lib/libkrb5.so,/usr/lib/libgssapi_krb5.so"`.

On AIX, the `dll-path` should include the archive file, if applicable, for example, `"<path>/libgssapi_krb5.a(libgssapi_krb5.a.so)"`. The `archive(shared_object)` syntax is not necessary if the library is a shared object or has been extracted from the shared object.

On Windows, the `dll-path` attribute is ignored. Tectia Server locates the correct DLL automatically.

The `allow-ticket-forwarding` attribute defines whether the server allows forwarding the Kerberos ticket over several connections. The attribute can have a value of `yes` or `no`. The default is `no`.

Note

On Microsoft Windows version 5.2 (Server 2003) and newer the possibility to allow Kerberos ticket forwarding is determined by the domain's Kerberos policy. For more information, see "How the Kerberos Version 5 Authentication Protocol Works".

Normally if a specified authentication method is not found on the server, the configuration file reading fails and the server will not restart. The `auth-gssapi` element may optionally take an `allow-missing` attribute, which can have a value of `yes` or `no`. If a value of `yes` is given for this attribute and GSSAPI plugin is not found during configuration reading, the server logs a warning to the syslog but will restart normally. The default is `no` (if GSSAPI is specified but not found, it is treated as fatal error and the server configuration reading fails).

Setting the `allow-missing` attribute to `yes` is useful when you want to use the same `ssh-server-config.xml` file on multiple servers and only some of the servers have Kerberos/GSSAPI available.

mapper

This element defines an external application to supplement authentication. Tectia Server uses the Tectia Mapper protocol (for more information, see Appendix E) to communicate with the external application.

The `mapper` element takes the following attributes: `command` (mandatory), `timeout` (optional), `chroot` (optional) and `user` (optional).

The `command` attribute specifies the command for running the external application.

The `chroot` attribute can be optionally used to define a directory where the mapper `user` is chrooted when the external application is run.

Caution

The external application will be launched under root privileges if mapper user is not specified.

You can use `timeout` to set the time limit for the external application to exit. The allowed value range is 1 to 3600 seconds, and the default value is 15 seconds. If the application hangs, Tectia Server will not kill it.

```
<authentication name="Zero_Trust_Authentication" action="deny">
 <auth-publickey />
 <authentication name="PrivX_User" action="allow">
  <selector>
   <certificate field="ca-list" pattern="PrivX_CA" />
   <certificate field="altname-upn" pattern="%username%" />
   <certificate field="extended-key-usage" pattern="ssh-client" />
  </selector>
  <authentication name="Additional_Mapper" action="allow">
   <mapper command="mapper_simple_example.py" chroot="/dir" user="mapper" />
  </authentication>
 </authentication>
</authentication>
```

Tectia Server sends the following data from its **blackboard** to the external application:

- `conn-feature-is-tectia`: Set to 'yes' if connecting client is Tectia based on version string.
- `conn-feature-strict-kex`: Set to 'yes' if strict key exchange has been used.
- `ip-fqdn`: The domain the client is connecting from
- `ip-addr`: The client's IP address
- `ip-port`: The port number the client is connecting from
- `interface-port`: The port the server is listening on
- `protocol-cipher-c2s`: Cipher algorithm used for encrypting data from client to server.
- `protocol-cipher-s2c`: Cipher algorithm used for encrypting data from server to client.
- `protocol-compression-c2s`: Compression algorithm used for data from client to server.
- `protocol-compression-s2c`: Compression algorithm used for data from server to client.
- `protocol-hostkey-algorithm`: Host-key algorithm used for the connection.

- `protocol-kex`: Key-exchange algorithm used for the connection.
- `protocol-mac-c2s`: MAC algorithm used for authenticating messages from client to server.
- `protocol-mac-s2c`: MAC algorithm used for authenticating messages from server to client.
- `session-id=1`: Session ID
- `user`: The user's name
- `user-name-no-domain`: The user's name without the domain part
- `original-user`: Original user's name. Useful if target user name is changed during authorization with set-user.
- `user-privileged`: If user has root privileges.
- `original-user-name-no-domain`: Original user's name without the domain part. Useful if target user name is changed during authorization with set-user.
- `original-user-privileged`: If original user has root privileges.
- `user-group`: User group
- `version-string`: The string sent by the client in version exchange

Additionally, when certificate authentication is used, also the following data is sent:

- `certificate-issuer-name`: The `Issuer Name` attribute read from the certificate
- `certificate-subject-name`: The `Subject Name` attribute read from the certificate
- `certificate-serial-number`: The `Serial Number` attribute read from the certificate

For the authentication to succeed, the external application must return "`success`" and an exit status 0. For more information on the parameters allowed by Tectia Mapper Protocol, see Section E.1.

The authentication will fail if the defined external application does not exist, or is killed by timeout, or if it returns an exit value other than 0.

For examples of how to use the `mapper` element in authentication, see Section 5.11.1 and Section 5.11.2.

authentication

The `authentication` elements can be nested within each other. The method(s) in the child element(s) must be passed in addition to the method in the parent element.

Authentication Examples

A sample `authentication-methods` element is shown below:

```
<authentication-methods>
  <authentication>
    <selector>
      <user-group name="staff" />
    </selector>
    <auth-publickey authorized-keys-directory="%IG/ssh2_authorized_keys" />
    <auth-password />
  </authentication>
</authentication-methods>
```

In this simple authentication example, the users who belong to group "staff" are allowed to log in using either public-key or password authentication. The user public keys are checked from an alternate location. As there are no other authentication blocks, all users that do not match to the selector are implicitly denied authentication.

Another sample `authentication-methods` element is shown below:

```
<authentication-methods>
  <authentication action="deny">
    <auth-publickey />
    <authentication action="allow" set-group="local-user">
      <selector>
        <ip address="10.1.55.14-10.1.55.99" />
        <user name="johnd" />
        <user name="janed" />
      </selector>
      <auth-keyboard-interactive max-tries="4">
        <submethod-radius>
          <radius-server address="10.1.61.128">
            <radius-shared-secret file="&configdir;/radius-secret-file"/>
          </radius-server>
        </submethod-radius>
      </auth-keyboard-interactive>
    </authentication>
    <authentication action="allow" set-group="finance-inspector">
      <selector>
        <user-group name="finance" />
      </selector>
      <auth-password />
    </authentication>
  </authentication>
</authentication-methods>
```

In the example above, all users are first required to authenticate using public-key authentication. Based on selector matching, also a second method needs to be passed (RADIUS via keyboard-interactive or password). A group is set based on the matched and passed authentication methods. If a user does not match to either of the child authentication elements, access is denied (the parent authentication element has the action set to `deny`).

If the action of the parent authentication element would be `allow`, the non-matching users would be let in after having passed public-key authentication.

See Section 5.12 for more examples of configuring authentication chains.

A sample `authentication-methods` element that sets requirements for certificate authentication is shown below:

```
<authentication-methods>
  <authentication action="allow">
    <auth-publickey />
    <authentication action="allow" set-group="admin">
      <selector>
        <user-privileged value="yes" allow-undefined="yes" />
        <certificate field="ca-list" pattern="exa-ca1,exa-ca2"
                     allow-undefined="yes" />
        <certificate field="subject-name" pattern="C=FI,
                      O=SSH, OU=*, CN=%username%"
                     ignore-suffix="yes" allow-undefined="yes" />
      </selector>
    </authentication>
    <authentication action="allow">
      <selector>
        <publickey-passed length="2048-4096" />
      </selector>
    </authentication>
    <authentication action="deny" />
  </authentication>
</authentication-methods>
```

In the example above, privileged users (administrators) are required to pass certificate authentication and the certificate must contain the correct fields. Other users are allowed to log in using a plain public key of a size from 2048 to 4096 bits.

In this example, the `allow-undefined` attribute has to be used in the selectors of the first nested authentication block. Otherwise, the authentication will end in error for users with plain public keys. When the user uses a plain public key, the server will not have the certificate fields to be matched defined. For more information, see the section called “Selectors and Undefined Data”.

Note

Specifying an explicit `deny` action last is necessary in a restrictive policy, as otherwise a non-matching connection would use the `allow` action of the parent element (if it passed public-key authentication with any key length). A better way to achieve the same result is to set the action of the parent authentication element to `deny` (as in the previous example).

The following example configuration shows how to define explicitly the certificate types that match the authentication policy:

```
<authentication-methods>
  <authentication action="deny">
    <auth-publickey />
    <authentication action="allow">
      <selector>
        <certificate field="extended-key-usage"
                     pattern="clientAuth,ssh-client"
                     explicit="yes" />
```

```
      </selector>
    </authentication>
  </authentication>
</authentication-methods>
```

This example configuration denies public-key authentication, and accepts only certificates that include either one or both of the `clientAuth` and `ssh-client` key types. `explicit="yes"` defines that the specified key purpose ID must be matched in the certificate, and so certificates with `anyExtendedKeyUsage` (or a missing key purpose ID) will not match.

The `services` Block

The `services` block defines the policy for the various services the server offers.

The `services` block contains one or more rules (`rule`) and optionally defines groups (`group`).

group

Creates a group that can be used as a basis for restricting services. Groups are defined based on selectors.

The `name` must be given as an attribute. The value of `name` must be a valid XML name beginning with a letter and containing alphanumeric characters or the underscore character without any whitespace.

If the user was already put to a group during authentication using the `set-group` attribute, the group definitions in the `services` element are ignored.

selector

The selectors define the users that belong to the group. The same selectors can be used as in the `authentication-methods` block. See Section 4.2.2 and the section called "The `authentication-methods` Block".

Sample `group` elements are shown below:

```
<!-- Remote access. -->
<group name="remote-access">
  <selector>
    <interface address="192.0.2.62" />
  </selector>
</group>

<!-- Backup. -->
<group name="backup">
  <selector>
    <user name="backup" />
  </selector>
</group>

<!-- Password change needed. -->
<group name="passwd-change">
  <selector>
```

```
    <user-password-change-needed />
  </selector>
</group>
```

rule

This element defines a rule for the specified `group` of users. Rules can be used to restrict the services and commands the server allows to the users. The element can have three attributes: `group`, `idle-timeout`, and `print-motd`.

The rules are read in order, and the first rule that matches the user's `group` is used. The match must be exact. No wildcards are allowed in the `group` attribute. If no `group` is specified, the rule matches to all users.

The `idle-timeout` attribute sets the idle timeout limit in seconds. If the connection (all channels) has been idle this long, the connection is closed. The default is `0` (zero), which disables idle timeouts.

The `print-motd` attribute defines whether the message of the day (`/etc/motd`) is printed when a user logs in interactively to a Unix server. The value must be `yes` or `no`. The default is `yes`.

Each rule can contain `environment`, `terminal`, `subsystem`, `tunnel-agent`, `tunnel-x11`, `tunnel-local`, `tunnel-remote`, and `command` elements.

An empty rule allows the specified group to perform all actions.

Note

The default (unnamed) rule allows all users access to all services. Keep the default rule as the last rule, so it will match to users that are not set in any group. Remember to edit the rule according to your security policy.

environment

This element defines the environment variables the user group is allowed to set at the client side. The variables are given in the `allowed` attribute as a comma-separated list. By default, the user can set the `TERM`, `PATH`, `TZ`, `LANG`, and `LC_*` variables.

Do **not** use * (asterisk), as it will allow any and all variables, and that can be a security risk.

Allowed variables are normally matched case-insensitively. If case-sensitive variables are needed, specify them using the `allowed-case-sensitive` attribute.

terminal

This element defines whether terminal access is allowed or denied for the user group. The word `allow` or `deny` can be given as the value of the `action` attribute. By default, terminal access is allowed.

On Unix systems, the `chroot` attribute can be optionally used to define a directory where the user is chrooted during the terminal session. For more information on chrooting, see Section 6.1.3.

If terminal access is denied, also shell commands are denied, unless commands are explicitly allowed or set as forced by the `command` element.

subsystem

This element defines a subsystem. The element can take the following attributes: `type`, `action`, `audit` (optional), `exec-directly` (optional), `application` (optional), and `chroot` (optional).

The `type` attribute must be given. It defines the subsystem, for which the settings are made. For example `sftp`.

The `action` attribute defines whether the use of the subsystem is allowed or denied. The possible values are `allow` and `deny`. The default is `allow`.

Note

Denying the SFTP subsystem denies both SFTP and SCP2 **scp2/scpg3** operations to the server, but it does not deny legacy OpenSSH-style SCP operations. To deny OpenSSH SCP (version 8 or older), you should restrict remote commands. See also `command` .

The optional `audit` attribute can be used to define whether the audit messages of the subsystem are recorded in the system log. Possible values are `yes` and `no`. The default is `yes`. The `audit` attribute can be used only with the SFTP subsystem.

The `exec-directly` attribute is only applicable to the `sft-server-g3` subsystem on Unix. The default value is `yes`, which means that the server will launch `sft-server-g3` directly without invoking the user's shell. Note that this will allow the user to run file transfers even if a dummy shell, such as `/bin/no-shell`, is specified in the user account. When the value is `no`, the server launches the user's shell which then executes `sft-server-g3`.

All other subsystem applications are run as if `exec-directly="no"` would be specified. For these subsystems it is not allowed to specify `exec-directly="yes"`.

An example configuration:

```
<subsystem type="sftp"
           action="allow"
           audit="no"
           exec-directly="no" />
```

The optional `application` attribute can be used to define the executable of the subsystem. This attribute is not necessary with the SFTP subsystem if the SFTP binary is in the default location. Example setting:

An example configuration:

```
<subsystem type="sftp"
           application="sft-server-g3"
           action="allow" />
```

On Unix, the optional `chroot` attribute can be used to define a directory where the user is chrooted when running the subsystem. For more information on chrooting SFTP, see the section called “Chrooting SFTP”.

The `subsystem` element can contain multiple `attribute` child elements.

attribute

This element can be used to define attributes for a subsystem.

The `attribute` element takes two attributes: `name` (mandatory) and `value` (optional).

SFTP subsystem attributes 'debug' with value '<level>' and 'debug-file' with value '<filename>' can be used to enable debugging. Variable '%U' expands to username and '%H' to hostname. Note that the SFTP user needs to have write permissions to the file, for example:

```
<subsystem   type="sftp"
             application="sft-server-g3">
  <attribute name="debug"
             value="8" />
  <attribute name="debug-file"
             value="/tmp/sftp_debug_%U.txt" />
</subsystem>
```

SFTP subsystem 'disable' attribute can be used to enforce download and upload restrictions such as deny 'write', 'read' and/or 'update' file operations, 'streaming' to use traditional SFTP instead of streaming, 'intrusive' to disallow rename, remove, mkdir, rmdir, setstat, symlink operations and 'dirlist' (directory listing) and 'mmclist' (MVS master catalog listing) operations. For example to prevent upload so that user is not allowed to open files for writing, remove or rename files or directories, or modify the contents of the filesystem in any other way:

```
<subsystem   type="sftp"
             application="sft-server-g3">
  <attribute name="disable"
             value="write" />
</subsystem>
```

On Windows platforms, UTF-8 mode is enabled by default for the SFTP subsystem. To use system's code page instead for file names on the server side, you can disable the 'utf8-mode' attribute. The following settings can be used to set the user home directory and virtual folders for the SFTP subsystem:

```
<subsystem type="sftp" application="sft-server-g3">
  <attribute name="utf8-mode" value="yes" />
  <attribute name="home" value="%USERPROFILE%" />
</subsystem>
```

For more information on virtual folders, see the section called “Defining SFTP Virtual Folders (Windows)”.

On Unix, you can use the `attribute` element to define a umask for the SFTP subsystem to overwrite the default file mode creation mask for the SFTP server. The value defined in the server configuration file takes precedence over any other umask settings, for example any umask settings in `.profile` or other shell init files.

Define the value for the umask in octal format (`0nnn`) or in decimal format (`nnn` without the leading zero). For example, the following setting defines the umask for all users to be 0022:

```
<subsystem type="sftp" application="sft-server-g3">
  <attribute name="umask" value="0022" />
</subsystem>
```

If you want to set different umasks for specific users or user groups, define the umask in a rule for the user or user group, for example:

```
<services>
  <group name="sftusers">
    <selector>
      <user name="jsmith" />
    </selector>
  </group>
  <rule group="sftusers">
    <subsystem type="sftp" action="allow" application="sft-server-g3">
      <attribute name="umask" value="0022" />
    </subsystem>
  </rule>
</services>
```

Note

If no umask is defined in the server configuration file, the umask that will be used depends on the user's client and shell as follows:

- When a user copies files using **scpg3**, **sftpg3**, or OpenSSH SFTP and `exec-directly` is set to `"yes"` (as it is by default), the server will launch **sft-server-g3** directly without invoking the user's shell, and the umask is inherited from the root user.

  ### Caution

  Support for legacy OpenSSH SCP in Tectia Server is not implemented via the SFTP subsystem but using a command called **scp1-compat-srv**. When a client uses OpenSSH version 8 or older SCP to connect to Tectia Server, the `umask` setting does not apply.

- When a user copies files using **scpg3**, **sftpg3**, or OpenSSH SFTP and `exec-directly` is set to `"no"`, the server launches the user's shell which then executes **sft-server-g3**. Depending on the user's shell, the umask of the running process is either inherited from the parent server process or taken from a client-side startup file (for example `/etc/profile` or `~/.bash_login`).

- When a user logs in to Tectia Server using a remote shell, a login shell is started and various client-side startup files may be executed. The umask can be set by the user in any of these files.
- When PAM authentication is used, you can set the umask in the PAM configuration file, for example:

  ```
  session    required     pam_umask.so umask=0022
  ```

  (For remote shell sessions, the umask settings in the startup files override the setting in the PAM configuration file.)

command

This element defines a shell command as allowed, denied, or forced. The element can have five attributes, however, all of them are not used at the same time: `action`, `interactive`, `application`, `application-case-sensitive`, and `chroot`.

The value of the `action` attribute can be either `allow`, `deny`, or `forced`. The default is `allow`.

If the `deny` action is set, all shell commands are denied and no further attributes should be specified. Commands are also denied if terminal access is denied in the rule and the command element is omitted.

For the `allow` action, the `application` can be optionally specified as an attribute. When the `allow` action is set and the `application` attribute is specified, running the specified application(s) is allowed and all other applications are implicitly denied. If the application is not given, running all commands is allowed.

For the `forced` action, the `application` must be given as an attribute. When the `forced` action is set, the specified application is run automatically when the user logs in successfully, instead of the application the user is trying to run. All other applications are implicitly denied.

For the `forced` action, the `interactive` can be given as an attribute. If the application that is run as forced requires user interaction, set the `interactive` attribute to yes. If the application that is run as forced does not require user interaction, set the `interactive` attribute to no. By default its value is set to no. This attribute is for Windows only.

If the `terminal` element is omitted from the rule and the `command` element specifies a forced command, terminal is implicitly denied. If the user requests a shell, the forced command is run instead.

If the `terminal` is explicitly allowed in the rule, the `forced` action of the `command` element applies only when the user tries to run remote commands. If the user requests a shell, he can get it normally and the forced command is not run.

If the SFTP `subsystem` is allowed, the user can also use the **scp2/scpg3** and **sftp2/sftpg3** programs normally. However, if the SFTP subsystem is denied, trying to use it will not cause the forced command to be run, but gives an error message.

Note

Support for legacy OpenSSH SCP in Tectia Server is implemented using a command called **scp1-compat-srv**. When a client uses OpenSSH version 8 or older SCP to connect to Tectia Server, the server invokes this command. Restrictions on remote commands apply also to OpenSSH-style SCP operations to the server.

Users can also define forced commands for public keys in their `authorization` files or OpenSSH-style `authorized_keys` files. However, if a command is defined in the `ssh-server-config.xml` file, it overrides any commands defined in the `authorization` or `authorized_keys` files. For more information, see the section called “Authorization File Options”.

Applications are normally matched case-insensitively. Alternatively, the application can be specified using the `application-case-sensitive` attribute.

On Unix systems, the `chroot` attribute can be optionally used to define a directory where the user is chrooted when running the command. For more information on chrooting, see Section 6.1.3.

tunnel-agent

This element defines whether agent tunneling (forwarding) is allowed or denied by the server.

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, agent forwarding is allowed.

For more information on agent forwarding, see Section 8.5.

tunnel-x11

This element defines whether X11 tunneling (forwarding) is allowed or denied by the server.

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, X11 forwarding is allowed.

For more information on X11 forwarding, see Section 8.4.

tunnel-local

This element defines a rule for local TCP tunnels (port forwarding). There can be several of these rules. When a user attempts tunneling, the rules are read in order and the first matching rule is used. For details, see the section called “Tunneling Rule Processing”.

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, local tunnels are allowed.

Tunneling restrictions can be further defined with the `src`, `dst` and `mapper` elements. Note that in each rule, you can define either `src` and/or `dst`, or `mapper`; you cannot use `mapper` in the same rule with `src` and `dst`.

src

This selector element specifies source address(es) and FQDN(s) for local tunnels.

Define the pattern to be matched with attribute `address` or `fqdn` or `fqdn-regexp`, but do not use them together.

The `address` can be in one of the following formats:

- 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 `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns. These patterns may also contain "*" and "?" globbing characters.

In the `fqdn-regexp` attribute, you can define a regular expression to match a range of FQDNs. Regular expressions follow the egrep syntax.

Note that SSH clients may spoof their source address, making this method somewhat unreliable in environments with untrusted clients.

tunnel-src

The **tunnel-src** matches tunnel-source addresses. In other words this matches to the address from which the tunnel starts. In scenarios such as chained tunneling, this may differ from the **src** reported by clients using the tunnel.

Define the pattern to be matched with attribute `address` or `fqdn` or `fqdn-regexp`, but do not use them together.

The `address` can be in one of the following formats:

- 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 `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns. These patterns may also contain "*" and "?" globbing characters.

In the `fqdn-regexp` attribute, you can define a regular expression to match a range of FQDNs. Regular expressions follow the egrep syntax.

dst

This element defines destination address(es) and port(s) for local tunnels.

The `address` or the `fqdn` (not both) can be given as an attribute. Also the `port` can be given.

The `address` can be in one of the following formats:

- 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 `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns. These patterns may also contain "*" and "?" globbing characters.

In the `fqdn-regexp` attribute, you can define a regular expression to match a range of FQDNs. Regular expressions follow the egrep syntax.

The `port` attribute can specify a single port or a port range (for example, `2000-9000`).

mapper

This element defines the use of an external application to verify information for local tunneling connections. Tectia Server uses the Tectia Mapper protocol (for more information, see Appendix E) to communicate with the external application.

The `mapper` element takes two attributes: `command` (mandatory) and `timeout` (optional).

The `command` attribute specifies the external application which is the executable of the subsystem.

Caution

The external application will be launched under administrator (root) privileges.

You can use `timeout` to set the time limit for the external application to exit. The allowed value range is 1 to 3600 seconds, and the default value is 15 seconds. If the application hangs, Tectia Server will not kill it.

Tectia Server sends the following data to the external application:

`user=`*`userid`*`:`*`username`*

Specifies the user id and user name

`user-privileged=true|false`

Specifies whether the user has administrator privileges.

`{tunnel-src}addr-ip=` *`ip-address`*

Specifies the tunnel's source IP address.

`{tunnel-src}port=` *`port`*

Specifies the tunnel's source port.

`{tunnel-src}addr-fqdn=` *`FQDN`*

Specifies the tunnel's source host (fully qualified domain name).

`{tunnel-dst}addr-ip=` *`ip-address`*

Specifies the tunnel's destination IP address.

`{tunnel-dst}port=` *`port`*

Specifies the tunnel's destination port.

`{tunnel-dst}addr-fqdn=` *`FQDN`*

Specifies the tunnel's destination host (fully qualified domain name).

For more information on local tunnels, see Section 8.2. For examples on using the local tunneling rules in the `ssh-server-config.xml` file, see Section 8.2.1.

tunnel-remote

This element defines a rule for remote TCP tunnels (port forwarding). There can be several of these rules. When a user attempts tunneling, the rules are read in order and the first matching rule is used. For details, see the section called “Tunneling Rule Processing”.

The word `allow` or `deny` can be given as the value of the `action` attribute. By default, remote tunnels are allowed.

Tunneling restrictions can be further defined with the `src` and `listen` elements.

src

This selector element specifies source address(es) and port(s) for remote tunnels.

Define the pattern to be matched with attribute `address` or `fqdn` or `fqdn-regexp`, but do not use them together.

The `address` can be in one of the following formats:

- 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 `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns. These patterns may also contain "*" and "?" globbing characters.

In the `fqdn-regexp` attribute, you can define a regular expression to match a range of FQDNs. Regular expressions follow the egrep syntax.

tunnel-dst

This selector element specifies the destination addresses for remote tunnels.

Define the pattern to be matched with attribute `address` or `fqdn` or `fqdn-regexp`, but do not use them together.

The `address` can be in one of the following formats:

- 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 `fqdn` attribute matches to an FQDN pattern (case-insensitive). The attribute can include a comma-separated list of allowed FQDN patterns. These patterns may also contain "*" and "?" globbing characters.

In the `fqdn-regexp` attribute, you can define a regular expression to match a range of FQDNs. Regular expressions follow the egrep syntax.

listen

This element defines listen address(es) and port(s) for remote tunnels.

The `address` and the `port` can be given as attributes.

The `address` can be in the formats described above for the `src` element.

The `port` attribute can specify a single port or a port range (for example, `2000-9000`).

For more information on remote tunnels, see Section 8.3.

Sample `rule` elements are shown below:

```
<!-- Administrators are allowed to do anything. -->
<rule group="admin" />

<!-- The finance inspector. -->
<rule group="finance-inspector" print-motd="no">
  <tunnel-local action="allow">
    <!-- Microsoft SQL ports. -->
    <dst fqdn="finance-db.example.com" port="1433" />
    <dst fqdn="finance-db.example.com" port="1434" />
  </tunnel-local>
  <tunnel-remote action="deny" />
  <!-- Can execute commands and shells, as no overriding behavior is defined. -->
</rule>

<!-- Remote access. -->
<rule group="remote-access" idle-timeout="600">
  <!-- Setting terminal action to "deny" also denies shell commands,
       unless they are specifically allowed.     -->
  <terminal action="deny" />
```

```
  <subsystem type="sftp" application="sft-server-g3" chroot="%homedir%" />
  <!-- The listed local tunnels are allowed, other local tunnels are
       denied. -->
  <tunnel-local action="allow">
    <!-- IMAP. -->
    <dst fqdn="imap.example.com" port="143" />
    <dst fqdn="imap.example.com" port="993" />
    <!-- POP. -->
    <dst fqdn="mail.example.com" port="109" />
    <dst fqdn="mail.example.com" port="110" />
    <dst fqdn="mail.example.com" port="995" />
  </tunnel-local>
  <tunnel-remote action="deny" />
</rule>

<rule group="backup">
  <terminal action="deny" />
  <!-- This account is only used to back up the disk drive. -->
  <command application="dd if=/dev/hda" action="forced" />
  <tunnel-local action="deny" />
  <tunnel-remote action="deny" />
</rule>

<!-- This rule is used to force password change. -->
<rule group="passwd-change">
  <terminal action="deny"/>
  <subsystem type="sftp" application="sft-server-g3" action="deny" />
  <command application="/usr/bin/passwd" action="forced" />
  <tunnel-local action="deny" />
  <tunnel-remote action="deny" />
</rule>

<!-- The default rule, used if the user has not been put to any group. -->
<rule>
  <!-- The listed environment variables are allowed and all others are
       denied. There is no "denied" setting. -->
  <environment allowed-case-sensitive="TERM,PATH,TZ,LANG,LC_*" />
  <terminal action="deny" />
  <subsystem type="sftp" application="sft-server-g3" chroot="%homedir%" />
  <!-- Only the date command is allowed. Other commands are denied. -->
  <command application="date" action="allow" />
  <tunnel-local action="allow" />
  <tunnel-remote action="deny" />
</rule>
```

Forcing Password Change

On Unix, Tectia Server enforces the changing of expired passwords.

Tectia Server implicitly adds the following group and rule at the top of the rules and groups in the configuration file:

```
<group name="passwd-change">
  <selector>
```

```
    <user-password-change-needed />
  </selector>
</group>

<!-- This rule is used to force password change. -->
<rule group="passwd-change">
  <terminal action="deny"/>
  <subsystem type="sftp" application="sft-server-g3" action="deny" />
  <command application="/usr/bin/passwd" action="forced" />
  <tunnel-local action="deny" />
  <tunnel-remote action="deny" />
</rule>
```

Therefore, when users whose passwords have expired attempt to log in, the settings in `<rule group="passwd-change">` are applied to them. The `passwd-change` group settings deny all services and force the user to enter a new password.

Note that the user needs an SSH terminal client to be able to change the password.

You can overwrite the implicitly added handling of expired passwords by defining a group named `passwd-change` and adding your own rule for it.

Note

Tectia Server enforces the changing of expired passwords also when the configuration file contains a group that is not named `passwd-change` but uses the `user-password-change-needed` selector. The contents of any rule specified for this group will not have an effect on the handling of expired passwords.

Note

In Tectia Server 6.4.9 and earlier it was possible to manually add rules for the handling of expired passwords, which made it possible to accidentally allow services for users after their password had expired. If you want the password handling to behave as in Tectia Server 6.4.9 or earlier, define a group named `passwd-change` with a selector that will never be true, for example:

```
<group name="passwd-change">
  <selector>
    <user name="invalid-user-name-does-not-exist" />
  </selector>
</group>
```

Add this group as the *last* `services` group in the configuration file and do not define any rules for the group.

This will prevent Tectia Server from implicitly adding a `passwd-change` group at the head of the services group list, allowing you to manually define your own rules for the handling of expired passwords.

On Windows, password change is handled differently than on Unix platforms, and it is not configurable. If a password change is required for the account by the server, user will be prompted to change the password

during authentication right after the validation of the old password. User will be logged on after successful password change.

Some third-party SSH clients may allow users to request password change themselves during authentication. In that case, it will be handled the same way as it would have been enforced by server.

Note

For accounts with empty password, and whose login is disabled by policy: "Accounts: Limit local account use of blank passwords to console logon only", the user will be prompted to change the password even when the user is not able to log on otherwise using password authentication (see *Empty/Blank Passwords* in Section 5.5).

Tunneling Rule Processing

The tunneling rules defined by the `tunnel-local` and `tunnel-remote` elements operate in the same way as selectors. Inside a tunnel rule, elements of different type are in AND relation to each other. If a tunnel rule contains several items of the same type, they are in OR relation to each other.

For example, the following rule matches if both the listen port and the source FQDN match:

```
<rule>
  <tunnel-remote action="allow">
    <listen port="1-9000" />
    <src fqdn="trusted.example.com" />
  </tunnel-remote>
...
</rule>
```

For example, the following rule matches if either of the source addresses match:

```
<rule>
  <tunnel-remote action="deny">
    <src address="192.168.23.1" />
    <src address="10.1.0.1" />
  </tunnel-remote>
...
</rule>
```

With several tunneling rules of the same type, the first matching rule is used. For example, the following configuration allows local tunnels to all other addresses in network *`192.168.14.0/24`* except *`192.168.14.21-192.168.14.30`*:

```
<rule>
  <tunnel-local action="deny">
    <dst address="192.168.14.21-192.168.14.30" />
  </tunnel-local>

  <tunnel-local action="allow">
    <dst address="192.168.14.0/24" />
  </tunnel-local>
...
</rule>
```

If the `tunnel-local` elements were in different order, tunnels to the whole *`192.168.14.0/24`* network would be allowed as any tunneling attempts would match the first (`allow`) rule and the second (`deny`) rule would not be read.

For more examples of tunneling rules, see Chapter 8.

Chapter 5 Authentication

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

The Tectia Server host can authenticate itself to the client using either public-key authentication or certificate authentication.

Different methods can be used to authenticate Secure Shell client users. These authentication methods can be used separately or combined, depending on the level of functionality and security you want. The server defines what methods are allowed, and the client defines the order in which they will be tried. The least interactive methods should be tried first. In case several interactive authentication methods are defined for user authentication, the client-side will alternate between the methods on each failed authentication attempt.

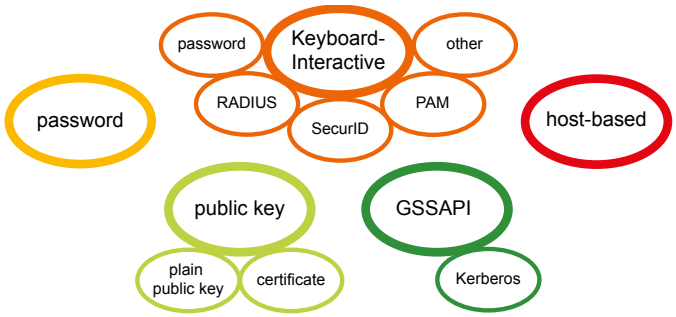


Figure 5.1. User authentication methods

Tectia Server allows GSSAPI, public-key, keyboard-interactive, and password in user authentication by default.

5.1 Supported User Authentication Methods

The following user authentication methods are supported in the Tectia client/server solution.

Table 5.1. User authentication methods supported by the Tectia client/server solution

Authentication method	Tectia Server		Tectia Client and ConnectSecure	
	Unix	Windows	Unix	Windows
Password[a]	x	x	x	x
Public-key	x	x	x	x
Certificate	x	x	x	x
Host-based	x	x	x	
Keyboard-interactive	x	x	x	x
PAM[b]	x		x	x
RSA SecurID[b]	x	x	x	x
RADIUS[b]	x	x	x	x
GSSAPI/Kerberos	x	x	x	x

[a] On SELinux enabled systems, password method uses PAM internally on the server side.

[b] Through keyboard-interactive.

5.1.1 Compatibility with OpenSSH Keys and Certificates

By default, the Tectia client/server solution uses private and public keys stored in the IETF standard Secure Shell v2 format. However, Tectia Client and Server can also use keys and related files in the legacy OpenSSH format or OpenSSH certificates.

The following OpenSSH-format keys are supported:

- server host key pair and host certificate pair
- trusted server host public keys, which clients use to authenticate servers
- user private keys (used by clients to authenticate to a server)
- authorized user public keys (used by a server to authenticate users), including public-key options
- OpenSSH user and host certificates
- OpenSSH CA-keys (used by a server to authenticate certificate users, or client to authenticate servers with host certificates)

5.2 Server Authentication with Public Keys

A public key is always created for Tectia Server during the installation phase. In addition, the Server administrator can generate more public-key pairs for the host, according to need.

The server is authenticated with a digital signature based on an RSA, DSA, ECDSA or Ed25519 public-key algorithm. At the beginning of each connection, the server sends its public key to the client for validation.

The key pair that the server uses in server authentication is defined in the server configuration file, `ssh-server-config.xml`, with the following elements:

```
<params>
  <hostkey>
    <private file="/etc/ssh2/hostkey" />
    <public file="/etc/ssh2/hostkey.pub" />
  </hostkey>
...
</params>
```

Giving the public key in the configuration file is not mandatory. It will be derived from the private key if it is not found otherwise. Specifying the public key will, however, decrease start-up time for the software, as deriving the public key is a somewhat time-consuming operation.

During the installation process, one RSA key pair (with the file names `hostkey` and `hostkey.pub`) is generated and stored in the `/etc/ssh2` directory on Unix and in the "`<INSTALLDIR>\SSH Tectia Server`" directory on Windows. By default, this key pair is used for server authentication.

Each Tectia Server can have multiple host keys. You could have, for example, the following set of parameters in your `ssh-server-config.xml` file:

```
<params>
  <hostkey>
    <private file="/etc/ssh2/hostkey_rsa" />
    <public file="/etc/ssh2/hostkey_rsa.pub" />
  </hostkey>

  <hostkey>
    <private file="/etc/ssh2/hostkey_dsa" />
    <public file="/etc/ssh2/hostkey_dsa.pub" />
  </hostkey>

  <hostkey>
    <private file="/etc/ssh2/hostkey_ecdsa" />
    <public file="/etc/ssh2/hostkey_ecdsa.pub" />
  </hostkey>
...
</params>
```

All keys are stored in memory when the **ssh-server-g3** process is started, which means that any one of them can be used to authenticate the server.

We recommend that you use a maximum of one key pair of each type (RSA, DSA, ECDSA, Ed25519). If also certificates are used in server authentication, an additional three host key pairs (RSA/ DSA/ECDSA with certificate) can be used for a total of seven host keys.

The host keys can be configured with the **Tectia Server Configuration** tool on the **Identity** page. See Section 4.1.6.

5.2.1 Generating the Host Key

A host public-key pair (3072-bit RSA) is always generated during the fresh installation of Tectia Server. You only need to regenerate it if you want to change your host key pair.

The command-line tool **ssh-keygen-g3** can be used to generate the host key pair. It can be used for creating the user key pairs as well.

On Unix, to (re)generate the host key, give the following command with root privileges:

```
# ssh-keygen-g3 -P -H hostkey
```

where:

`-P` indicates that the key has an empty passphrase

`-H` indicates that the key pair is stored in the default host key directory

On Windows, to (re)generate the host key, give the following command:

```
ssh-keygen-g3.exe -P -H hostkey
```

This will generate a 3072-bit RSA key pair (without a passphrase) and save it in the default host key directory (`/etc/ssh2` on Unix, "`<INSTALLDIR>\SSH Tectia Server`" on Windows) with the names `hostkey` and `hostkey.pub`. For more information on the key generation options, see ssh-keygen-g3(1).

After the new key pair has been created, run **ssh-server-ctl** to reconfigure the server. See ssh-server-ctl(8).

Note

The private key of the server must *never* be readable by anyone but `root` on Unix and by the **Administrators** group and the **SYSTEM** account on Windows. Store the private key in a safe directory where access is denied for all others.

5.2.2 Notifying the Users of Host Key Changes

Administrators that have other users connecting to their server should notify the users of any host key changes. The users will receive a warning the next time they connect because the host key the users have saved on their disk for your server does not match the host key now being actually provided by your server. The users may not know how to respond to this warning.

You can run `ssh-keygen-g3` to calculate the fingerprint of your new public host key and you can provide the fingerprint to your users via some unalterable method (for example, by a digitally signed e-mail or by displaying the fingerprint on a secured bulletin board).

On Unix, the command for calculating the fingerprint is:

```
# ssh-keygen-g3 -F hostkey.pub
```

On Windows, the command is:

```
ssh-keygen-g3.exe -F hostkey.pub
```

When the users connect and receive the error message about the host key having changed, they can compare the fingerprint of the new key with the fingerprint you have provided in your e-mail, and ensure that they are connecting to the correct Tectia Server. Inform your users to notify you if the fingerprints do not match, or if they receive a message about a host key change and do not receive a corresponding message from you notifying them of the change.

This procedure can help ensure that you do not become a victim of a man-in-the-middle attack, as your users will notify you if the host key fingerprints do not match.

It is also possible to send the public host key to the users via an unalterable method. The users can save the key in the `$HOME/.ssh2/hostkeys` directory on Unix or in the `%APPDATA%\SSH\HostKeys` directory on Windows as `key_<port>_<host>.pub` (for example, `key_22_banana.ssh.com.pub`). In this case, a manual fingerprint check is not needed.

5.2.3 Key rotation

Like passwords, host keys (and the authorizations they create) should be rotated regularly to limit their exposure to misuse.

Key rotation for server host keys can be set in the server configuration GUI under Identity tab (see Section 4.1.6), or it can be set in the `server-config.xml` `hostkey` element (see **hostkey**).

Once a rotation period has been set for a host key, a newly generated key will replace the old one when the key rotation period ends. A key can be set up with a rotation margin period, which is a time span before the rotation, during which the new key is generated, and advertised to clients. Advertising the new key before key rotation allows clients to be prepared for the changing of the host key. If no rotation period is set, the automatic key rotation is disabled.

The host keys can also be changed manually by generating a new key and/or editing an existing keys' path in the server configuration GUI.

Please see Appendix I for more details on what should be taken into consideration when changing the Host Key of Tectia Server.

5.3 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.

After the certificate has been created for a server, it can be enrolled to the client hosts.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 with 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 of the server. This check can be disabled by setting the `end-point-identity-check` attribute of the `cert-validation` element in the client configuration file (`ssh-broker-config.xml`) to `no`.

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. You can configure how often the CRL is refreshed from the repository. See *Tectia Client User Manual* for more information.

5.3.1 Certificate Enrollment Using `ssh-cmpclient-g3`

Certificates can be enrolled using the `ssh-cmpclient-g3` command-line tool (`ssh-cmpclient-g3.exe` on Windows).

To configure Tectia Server to authenticate itself using X.509 certificates, perform the following tasks:

1. Enroll a certificate for the server.

   This can be done with the `ssh-cmpclient-g3` command-line tool, for example:

   ```
   $ ssh-cmpclient-g3 INITIALIZE \
     -P generate://ssh2@rsa:3072/hostcert_rsa \
     -o /etc/ssh2/hostcert_rsa \
     -p 62154:ssh \
     -s "C=FI,O=SSH,CN=testserv;dns=testserv.ssh.com" \
     http://pki.ssh.com:8080/pkix/ \
     'C=FI, O=SSH Communications Security, CN=Secure Shell Test CA'
   ```

   Note that the DNS address parameter (`dns`) needs to correspond to the fully qualified domain name of the server.

   Remember to define also the SOCKS server (`-S`) before the CA URL, if required.

   For more information on the `ssh-cmpclient-g3` syntax, see ssh-cmpclient-g3(1).

2. Define the private key and the server certificate in the `ssh-server-config.xml` file:

   ```
   <params>
     <hostkey>
       <private file="/etc/ssh2/hostcert_rsa" />
       <x509-certificate file="/etc/ssh2/hostcert_rsa.crt" />
     </hostkey>
   ...
   </params>
   ```

   Alternatively, when using the **Tectia Server Configuration** tool, enter the private key and certificate filenames on the **Identity** page. See Section 4.1.6.

3. Run `ssh-server-ctl` to take the new configuration in use. See ssh-server-ctl(8).

   On Windows, just click **Apply** to take the new settings in use.

5.4 Server Authentication Using External Host Keys

In addition to conventional keys and certificates stored as files on disk, several external key providers are available for accessing keys and certificates stored in hardware tokens or external software modules.

The example below initializes the `software` external key provider, which is used to access keys and certificates on disk, and instructs it to read all keys in `/etc/ssh2/hostkeys`.

```
<params>
  <hostkey>
    <externalkey type="software"
      init-info="directory(/etc/ssh2/hostkeys)"/>
  </hostkey>
...
</params>
```

Each `hostkey` element can be used for setting up one external key provider. Each key provider may provide any number of keys to the server. It should be noted that due to the limitations of the SSH2 protocol, having more than one key of each type (RSA, DSA, ECDSA, Ed25519, X.509 certificate with RSA key, X.509 certificate with DSA key and X.509 certificate with ECDSA key) is discouraged.

For more information on the different external keys and their initialization strings, see **externalkey** in ssh-server-config(5).

5.5 User Authentication with Passwords

The password authentication method is set up by default, so it is easy to implement and requires no configuring. Since all communication is encrypted, passwords are not available for eavesdroppers.

On Windows, Tectia Server does not need a user management program of its own – the user accounts are created with the standard Windows User Manager.

Tectia Server will record a login failure for each failed password authentication attempt.

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

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.

To enable password authentication on the server, the `authentication-methods` element of the `ssh-server-config.xml` file must contain an `auth-password` element. For example:

```
<authentication-methods>
  <authentication action="allow">
    <auth-password failure-delay="2" max-tries="3" />
    ...
  </authentication>
</authentication-methods>
```

Also other authentication methods can be allowed.

By using selectors, it is possible to allow or require password authentication only for a specified group of users. For more information, see Section 4.2.2.

Using the **Tectia Server Configuration** tool, password authentication can be allowed on the **Authentication** page. See Section 4.1.12.

Note

Passwords can also be used as a submethod in keyboard-interactive authentication. For more information, see Section 5.9.1.

5.5.1 Expired Passwords

On Unix, Tectia Server enforces the changing of expired passwords. For more information, see the section called “Forcing Password Change”.

On Windows, password change is handled differently than on Unix platforms, and it is not configurable. If the server requires a password change for an account, the user will be prompted to change the password during authentication, right after the validation of the old password. The user will be logged on after a successful password change.

Some third-party SSH clients may allow users to request password change themselves during authentication. In that case, it will be handled the same way as it would have been enforced by server.

Note

For accounts with empty password, and whose logon is disabled by policy: "Accounts: Limit local account use of blank passwords to console logon only", the user will be prompted to change the password even when the user is not able to log on otherwise using password authentication.

5.5.2 Empty/Blank Passwords

Tectia Server allows users with empty passwords to log in by password authentication method.

On Windows, local users with empty password can be restricted to log on from a physical console only by using the security policy “Accounts: Limit local account use of blank passwords to console logon only”. If this policy is enabled (as it is by default), users with empty password cannot log on to Tectia Server using password authentication. However, the same users can still log on to Tectia Server using other authentication methods that do not involve using the account's password, for example public key authentication.

Note

The policy “Accounts: Limit local account use of blank passwords to console logon only” does not apply to domain accounts.

5.5.3 User Logon Rights on Windows

User login requires the rights to *log on locally* and *access this computer from the network*. On domain controllers, these rights are disabled by default. If Tectia Server has been installed on a domain controller, the log on locally and the access this computer from the network permissions must be enabled on the domain controller for the `Domain Users` group.

Tectia Server allows defining locally the user logon types that are allowed on the host. By default, the Windows-set logon types are used, but for password-based authentication methods you can define `windows-logon-type`. For XML configuration instructions, see **settings** . For Tectia Server Configuration GUI instructions, see Section 4.1.2.

For example, in case you need to enable accounts that do *`NOT`* have the right to log on locally, use setting `windows-logon-type="network"`.

5.6 User Authentication with Public Keys

Public-key authentication is based on the use of digital signatures and provides very good authentication security. To use public-key authentication, the user must first create a key pair on the client, and upload the public key to the server.

The default directory where Tectia Server stores the users' public keys is `$HOME/.ssh2/authorized_keys` on Unix, and `%USERPROFILE%\.ssh2\authorized_keys` on Windows. The directory can be changed with the `authorized-keys-directory` attribute in the `ssh-server-config.xml` file. See **auth-publickey** .

The user is required to have the `read` rights, (and optionally the `write` rights) to the public-key files and directories, but the locations must not be accessible to other users. The `read` permissions are required for the `key.pub` file, the `authorized_keys` directory, and to the `authorization` file, if used. The `write` permission to these files are needed if the users are allowed to upload their own keys to the server.

To enable public-key authentication on the server, the `authentication-methods` element of the `ssh-server-config.xml` file must contain an `auth-publickey` element. For example:

```
<authentication-methods>
  <authentication action="allow">
    <auth-publickey authorized-keys-directory="%D/.ssh2/authorized_keys" />
    ...
  </authentication>
</authentication-methods>
```

Also other authentication methods can be allowed.

By using selectors, it is possible to allow or require public-key authentication only for a specified group of users. See Section 4.2.2 for more information.

Using the **Tectia Server Configuration** tool, public-key authentication can be allowed on the **Authentication** page. See Section 4.1.12.

5.6.1 Using the Authorization File

Tectia Server 4.x (and earlier) required an authorization file that listed the user public keys that are authorized for login. Using the authorization file with Tectia Server 5.0 and later is optional. If the file does not exist, Tectia Server looks for authorized public keys in the `authorized-keys` directory (as described in Section 5.6 above), and if that fails, in the default directory for user public-keys.

The default location for the authorization file is `$HOME/.ssh2/authorization` on Unix, and `%USERPROFILE%\.ssh2\authorization` on Windows. The file location can be changed with the **authorization-file** attribute in the `ssh-server-config.xml` file. See **auth-publickey** .

The authorization file contains a list of public key filenames each preceded by the keyword `Key`. If there is more than one `Key`, they are all authorized for login. For more information on the syntax of the authorization file, see `$HOME/.ssh2/authorization` (user-specific) under the section called “Files”.

Tectia Client on Windows can upload the public keys and edit the authorization file automatically.

5.6.2 Using Keys Generated with OpenSSH

Tectia Server supports also user public keys generated with OpenSSH. The OpenSSH keys can be configured the same way as described above for keys generated with Tectia Client.

Alternatively, the OpenSSH-style authorized keys file can be specified in the `ssh-server-config.xml` file by using the `openssh-authorized-keys-file` attribute. See **auth-publickey** . An example configuration is shown below:

```
<authentication-methods>
  <authentication action="allow">
    <auth-publickey authorization-file="%D/.ssh2/authorization"
                    openssh-authorized-keys-file="%D/.ssh/authorized_keys" />
    ...
  </authentication>
</authentication-methods>
```

Tectia Server checks the file defined in `openssh-authorized-keys-file` if it cannot find a matching key in the Tectia `authorization-file` or the `authorized-keys-directory`. Public keys defined in the Tectia locations have precedence over the keys in the OpenSSH file if the same key is defined in both.

5.6.3 Special Considerations on Windows

On the Tectia Server for Windows, the recommended location for public keys is the `%USERPROFILE%\.ssh2` directory. This location reflects the standard Unix usage and works with the default settings of Tectia Client automatic key upload, and the user's profile directory always has the appropriate access permissions (set by the operating system during the account creation).

The user configuration directory can be changed on the **General** page of the **Tectia Server Configuration** tool. See Section 4.1.2.

If users need to manage their public keys themselves, the administrator should inform the users about the location of the user configuration directory. Otherwise, the administrator has to place the user's public keys in the proper directory.

Note

Tectia Client uses SFTP for the automatic uploading of the public key. It will not succeed if the user configuration directory has been set to a location that is not under the user's SFTP home directory. By default, both directories are under `%USERPROFILE%`.

If you want to enable automatic public-key upload for the users, change both the user configuration directory and the SFTP user home directory to point to the same directory. See the section called “SFTP”.

For example, set `D:\SFTP\%username%` as the SFTP user home directory and `D:\SFTP\%username%\.ssh2` as the user configuration directory.

See also the general considerations on user name handling in Section 5.5.3.

5.6.4 Authorized Keys on a Windows Network Drive

Tectia Server supports storing domain users' authorized keys (and authorization files) on a network drive.

Because the user's network drives are restored only after the user has been fully authenticated, the path to the authorized keys directory must be specified in UNC format, for example:

```
\\server.my-company.com\dfs\authorized_keys
```

Tectia Server is accessing the network drive from a thread impersonating the user who is logging on. The impersonation token is obtained via Microsoft Kerberos extension **Service for User to Self (S4U2self)**[1]. Therefore, the computer domain account is acting on behalf of the user.

To enable access to network resources this way, the following requirements must be met:

- The computer must be configured on a domain controller to be trusted for delegation of CIFS services for the particular network drive. For instructions on how to configure this, see Section 9.3.6.
- The individual users must have read access to their files granted in the access control list of the folder itself, as well as in the network drive.

Example configuration:

```
<authentication-methods login-grace-time="600">
  <authentication name="Default-Authentication" action="allow" password-cache="no">
    <auth-publickey
      authorized-keys-directory="\\server.my-company.com\dfs\authorized_keys\%U"
      require-dns-match="no" />
    <auth-password failure-delay="2" max-tries="3" />
    <auth-gssapi allow-ticket-forwarding="no" />
    <auth-keyboard-interactive failure-delay="2" max-tries="3">
      <submethod-password />
    </auth-keyboard-interactive>
  </authentication>
</authentication-methods>
```

[1] S4U2self is an extension that allows a service to obtain a Kerberos service ticket to itself. The service ticket contains the user's groups and can therefore be used in authorization decisions.

5.7 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 both 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 certificate authority (CA) (self-signed certificate) is enough.

In brief, certificate authentication works as follows:

1. The client sends the user certificate (which includes the user's public key) to the server. The packet also contains random data unique to the session and 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 with the rules in the server configuration file to decide whether login is allowed or not.

Compared to conventional public-key authentication, this method is more secure because the system checks that the user certificate was issued by a trusted CA. In addition, certificate authentication is more convenient because no local database of user public keys is required on the server.

It is also easy to deny a user's access to the system by revoking his or her certificate, although this does not take effect until the next CRL update and requires that every other authentication method has been disabled. The status of a certificate can be checked 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 used if the certificate contains a valid **Authority Info Access** extension or if an `ocsp-responder` has been defined in the `ssh-server-config.xml` file. If no OCSP responder is defined or the OCSP connection fails, CRLs are used. The certificate should contain a valid **CRL Distribution Point** extension or an LDAP server for CRL fetching should be defined in the `ssh-server-config.xml` file.

5.7.1 Configuring Certificates

To configure the server to allow user authentication with X.509 certificates, perform the following tasks:

1. Acquire the CA certificate and copy it to the server 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. Specify the CA certificate and the CRL and OCSP settings in the `ssh-server-config.xml` file. An example is shown below:

```
<params>
  ...
  <cert-validation socks-server-url="socks://fw.example.com:1080">
    <ldap-server address="ldap.example.com" port="389" />
    <ocsp-responder validity-period="60" url="https://ca.example.com/ocsp-1" />
    <cert-cache-file file="/var/cert-cache.dat" />
    <crl-auto-update update-before="30" minimum-interval="600" />
    <crl-prefetch interval="1800" url="http://ca.example.com/default.crl" />
    <dod-pki enable="no" />
    <ca-certificate name="exa-ca1" file="/etc/ssh2/exa-ca1.crt" />
  </cert-validation>
</params>
```

You can define several CA certificates by using several `ca-certificate` elements. The server will accept only certificates issued by the defined CA(s). Only the `ca-certificate` elements are mandatory, all other configuration items featured above are just examples that may be used as needed.

The SOCKS server must be defined if the OCSP and CRL (LDAP) services are located behind a firewall.

Using the **Tectia Server Configuration** tool, the corresponding settings can be made on the **Certificate Validation** page. See Section 4.1.9.

3. Certificate authentication is a part of the `publickey` authentication method. Enable public-key authentication in the `ssh-server-config.xml` file and create rules that specify which certificates authorize logging into which accounts.

   The following is an example of certificate authentication rules in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
    <auth-publickey />
    <authentication action="allow">
      <selector>
        <certificate field="ca-list" pattern="exa-ca1,exa-ca2" />
        <certificate field="issuer-name" pattern="C=FI, O=SSH, CN=*" />
        <certificate field="subject-name" pattern="C=FI, O=SSH, CN=%username%" />
        <certificate field="serial-number" pattern="123456" />
        <certificate field="altname-email" pattern="%username%@ssh.com" />
        <certificate field="altname-upn" pattern="%username-without-domain%@ssh" />
        </selector>
    </authentication>
    <authentication action="deny" />
  </authentication>
</authentication-methods>
```

   In this example, Tectia Server tries to match all certificates offered by the client against the `certificate` selectors. As the last action, access is denied for all users whose certificates were not explicitly allowed. This is not strictly needed, since the server automatically inserts an authentication block named `implicit-certificate-deny` after other blocks to catch all certificate authentications that do not match anything else.

In the example, users with normal public keys will cause the authentication to end in error because the `allow-undefined` attribute is not set. See also the section called “Authentication Examples”.

Certificate authentication can be restricted using the following `field` attributes:

- `ca-list`: The pattern is a comma-separated list of CA names. The names that are defined in the `ca-certificate` element are used.
- `issuer-name`: The pattern is the required certificate issuer name in LDAP DN (distinguished name) string format. The issuer name may contain glob patterns ('*' and '?') but only in the component values, not names. For example, `"C=FI, O=SSH, CN=*"` is a legal pattern, but `"C=FI, *=SSH, CN=TestCA"` is not).
- `subject-name`: The pattern is the required subject name in LDAP DN string format. Matching is done in similar manner as with the issuer name described above.
- `serial-number`: The pattern is the required serial number of the certificate. A combination of issuer name and serial number can be used to uniquely identify a certificate.
- `altname-email`: The pattern is the e-mail address that must be present in the certificate as a subject alternative name.
- `altname-upn`: The pattern is the principal name that must be present in the certificate as a subject alternative name.

The patterns of type `subject-name`, `issuer-name`, `altname-email` and `altname-upn` can also contain special strings which are processed before comparing the pattern with the user's certificate. These strings are `%username%` (user's login name), `%username-without-domain%` (Windows only, user's login name without the domain part), `%homedir%` (user's home directory), and `%hostname%` (the name of the host the user is logging from, reverse mapped from the IP).

Caution

When creating selector lists for the public-key method, make sure that every selector ties the user name to the certificate in some way, either by including a **User name** field, or by putting the special substitution string `%username%` or `%username-without-domain%` to a field used to match some field in the certificate. Failing to do this may cause unintended consequences, for example, authentication can succeed with many different user names with a single certificate.

Similarly, when creating selector lists for the host-based method, make sure that every selector has a field that ties the certificate to the client host, using the `%hostname%` special substitution string.

Using the **Tectia Server Configuration** tool, certificate authentication rules can be configured on the **Authentication** page. For instructions, see Section 5.7.2.

4. Run `ssh-server-ctl` to take the new configuration in use. See ssh-server-ctl(8).

Click **Apply** to take the new settings in use.

5.7.2 Configuring User Authentication with Certificates on Windows

To configure Tectia Server to allow user authentication with X.509 certificates, perform the following tasks using Tectia Server Configuration GUI:

1. Launch Tectia Server Configuration GUI.

   Select Start > All Programs > Tectia Server > **Tectia Server Configuration**.

2. Under **GUI Mode**, select **Advanced** to view all available options and groups.

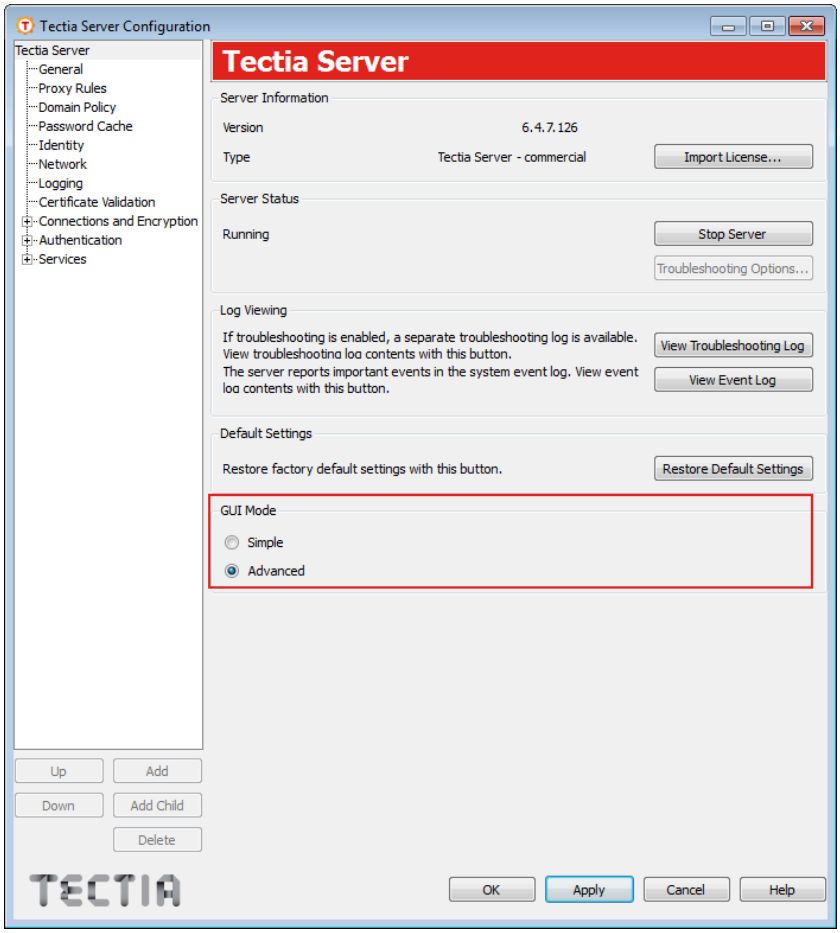


Figure 5.2. Selecting Advanced GUI mode

3. Go to the **Certificate Validation** page and select the **CA Certificates** tab.

4. Add the trust anchors and intermediate CA certificates that are needed for the certificate validation. Root CA certificates or intermediate CA certificates can be added as trust anchors. Normally you need to add only the CA certificate that can issue certificates for the users into Tectia Server configuration. That is, you need not create the whole trust path in the configuration.

Note

CA certificates are by default added to the **CA Certificates** list as trust anchors, meaning that revocation checks are not performed on them. When adding a new intermediate CA certificate, clear the **Trusted CA** check box to enable revocation checks.

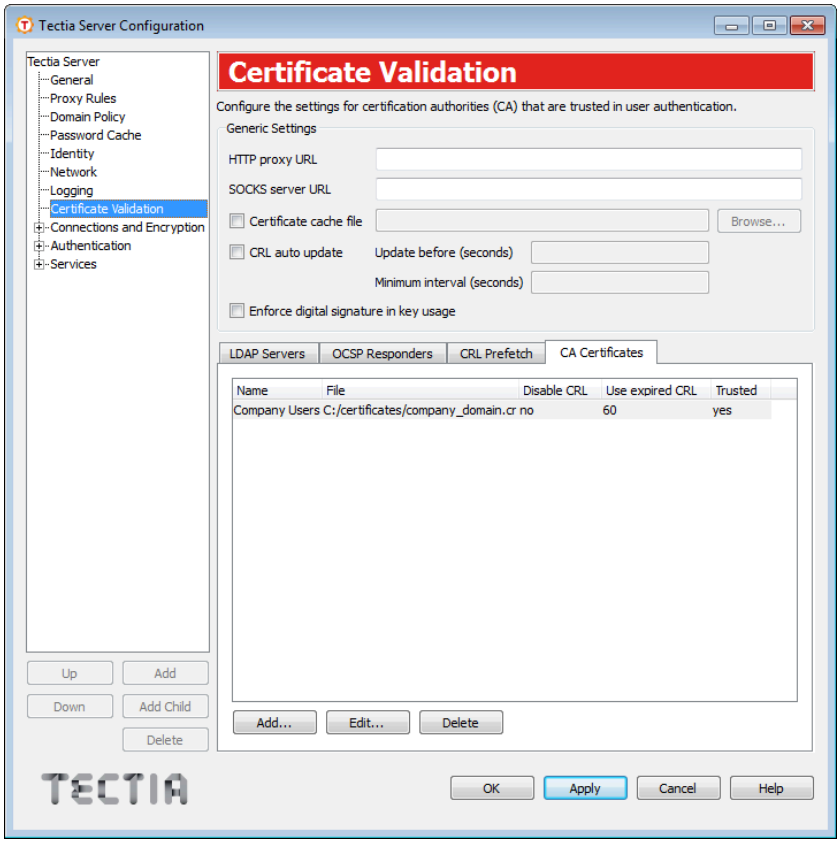


Figure 5.3. Adding CA certificates

Note

In case you have an LDAP server in use, you only need to add the root CA certificate into the server configuration. Tectia Server can retrieve the intermediate CA certificates that are issued by the root CA certificate automatically from the LDAP server. For example, if `Company Users` is added as a trust anchor and the intermediate CA certificates are stored in the LDAP, end entities certified by the root or intermediate CA certificates will be trusted.

For more information about certificate validation, see Section 4.1.9 .

5. Go to **Authentication** and select **Default Authentication** to configure selectors and parameters for the group. Note that this authentication group is available in the default configuration of Tectia Server.

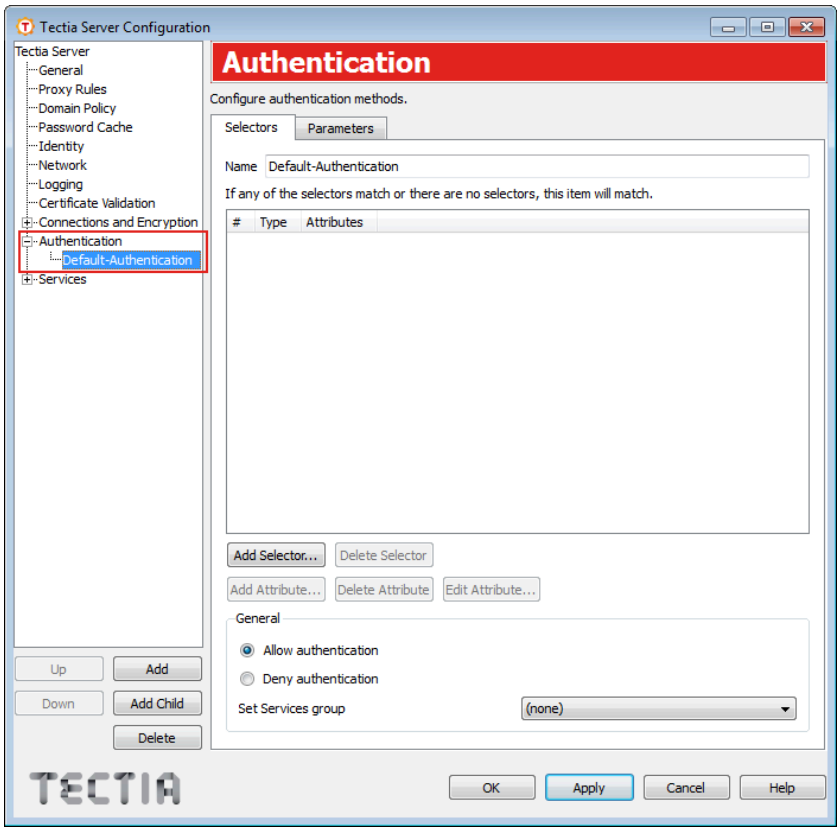


Figure 5.4. Creating authentication group

6. On the **Selectors** tab, enter a name for the authentication group.

   Leave the selectors list empty, all incoming users are selected into this authentication group and to the authentication method chain. This is the first authentication group that you need to create for the authentication method chain. There will be two authentication groups in the chain.

7. On the **Parameters** tab, make sure that the **Allow public-key authentication** option is selected.

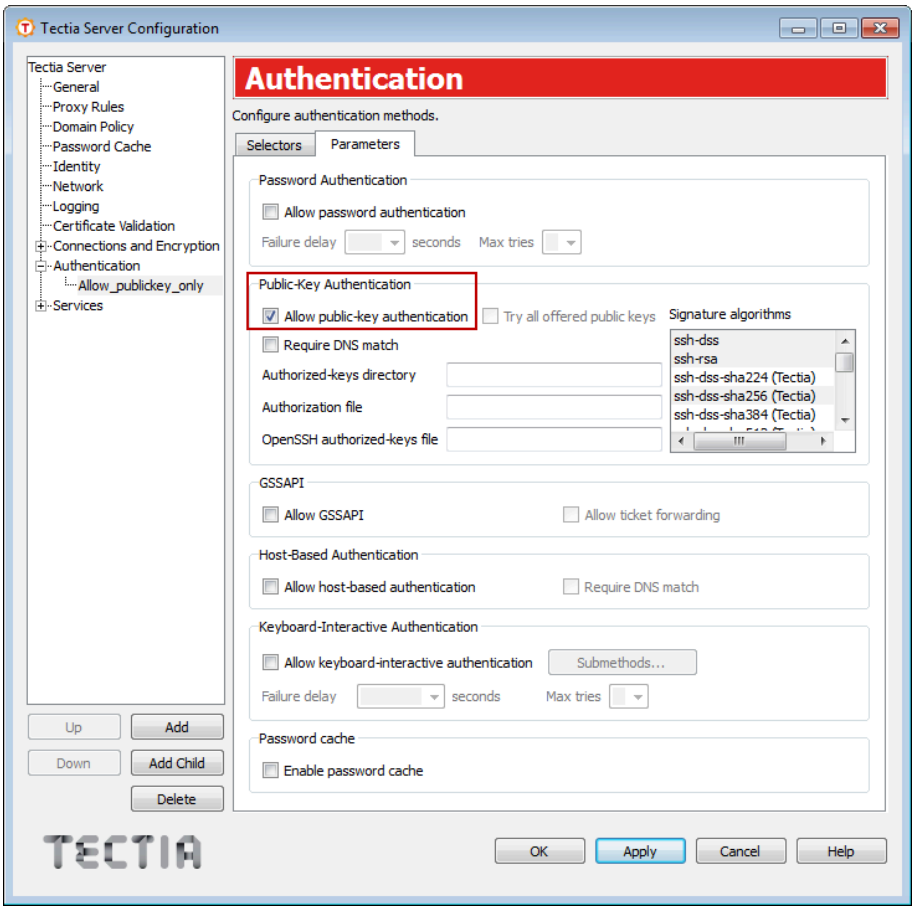


Figure 5.5. Allowing public key authentication

8. Create a child authentication group which will be used to check certain fields from the end user's certificate. That is, you are configuring your selector for the certificates. Click the **Add Child** button and enter a name for the child authentication group.

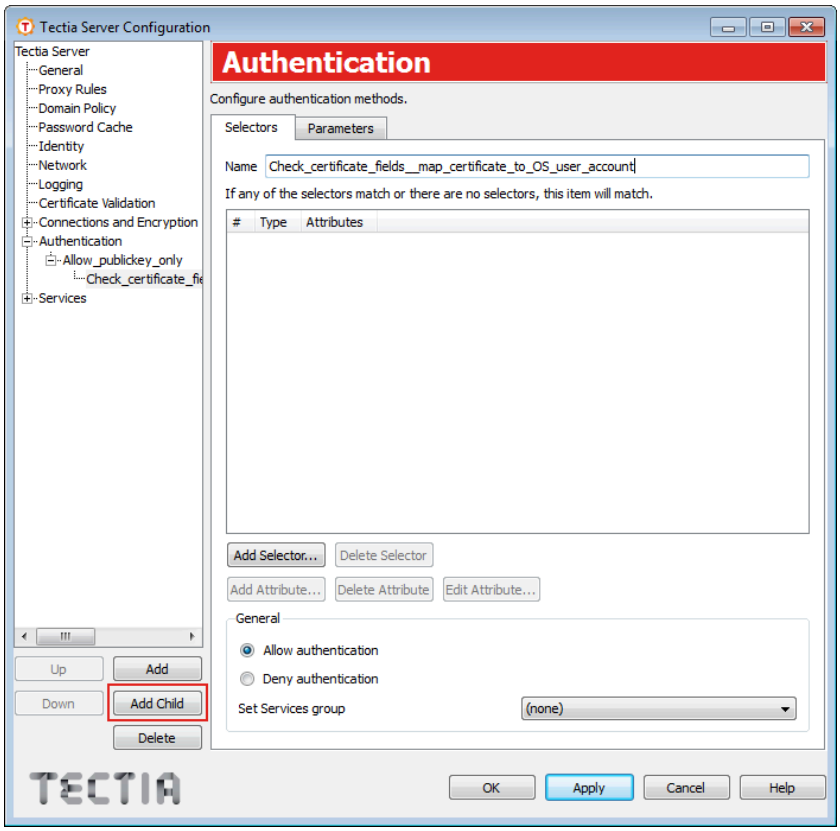


Figure 5.6. Creating a child authentication group

9. On the **Selectors** tab of the child authentication group, click the **Add Selector** button. From the list, select **Certificate** and click **OK**.

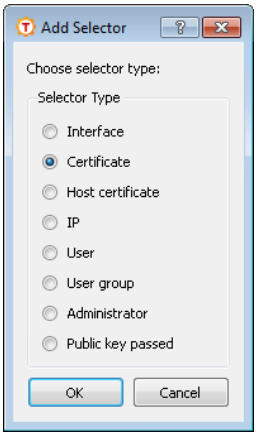


Figure 5.7. Adding a selector for a certificate

10. In the **Certificate Selector** dialog box, select which field on the certificate you wish to authenticate against.

11. Enter the pattern in the field.

It is extremely important to create a mapping between real OS user accounts and the end users' certificates so that a single end user can only access a single specific OS user account with their personal certificate and not all OS user accounts. For example, if you use **subject-name**, the pattern could be:

```
CN=%username-without-domain%, CN=USERS, DC=DEMO, DC=SSH, DC=COM
```

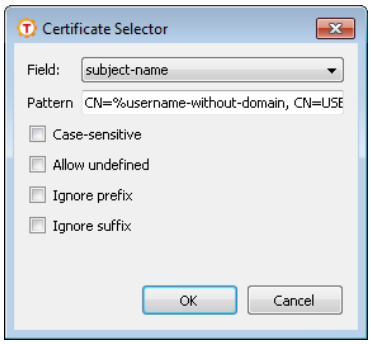


Figure 5.8. Entering a pattern for the certificate selector

12. Once you have made your changes, click **OK**.

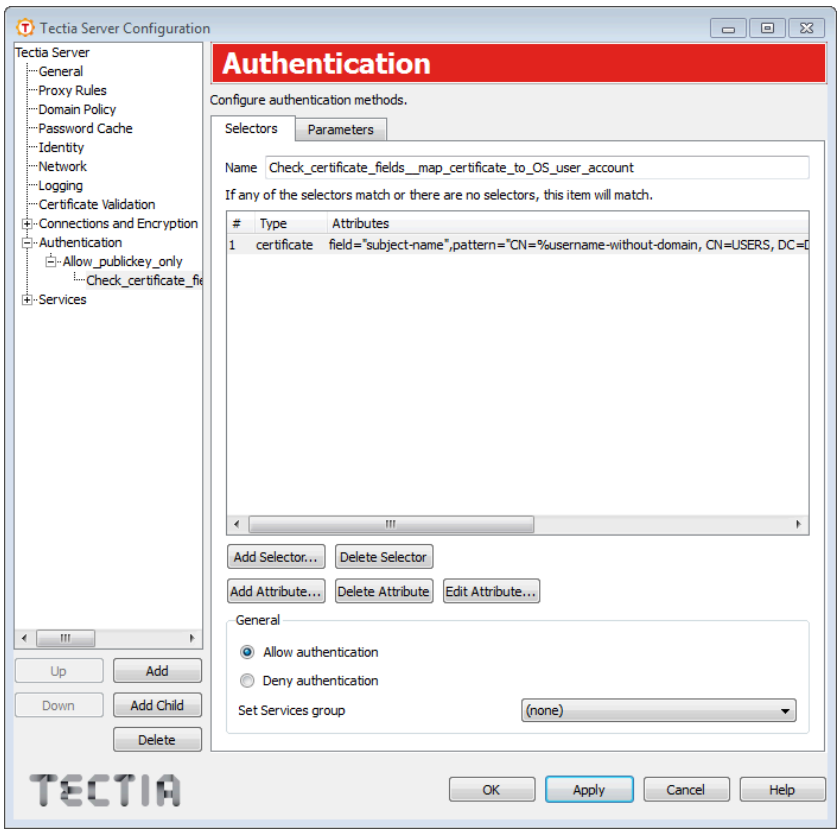


Figure 5.9. Completed selector

13. On the **Parameters** tab, unselect all authentication methods because the parent authentication group checks whether the public key authentication is successful.

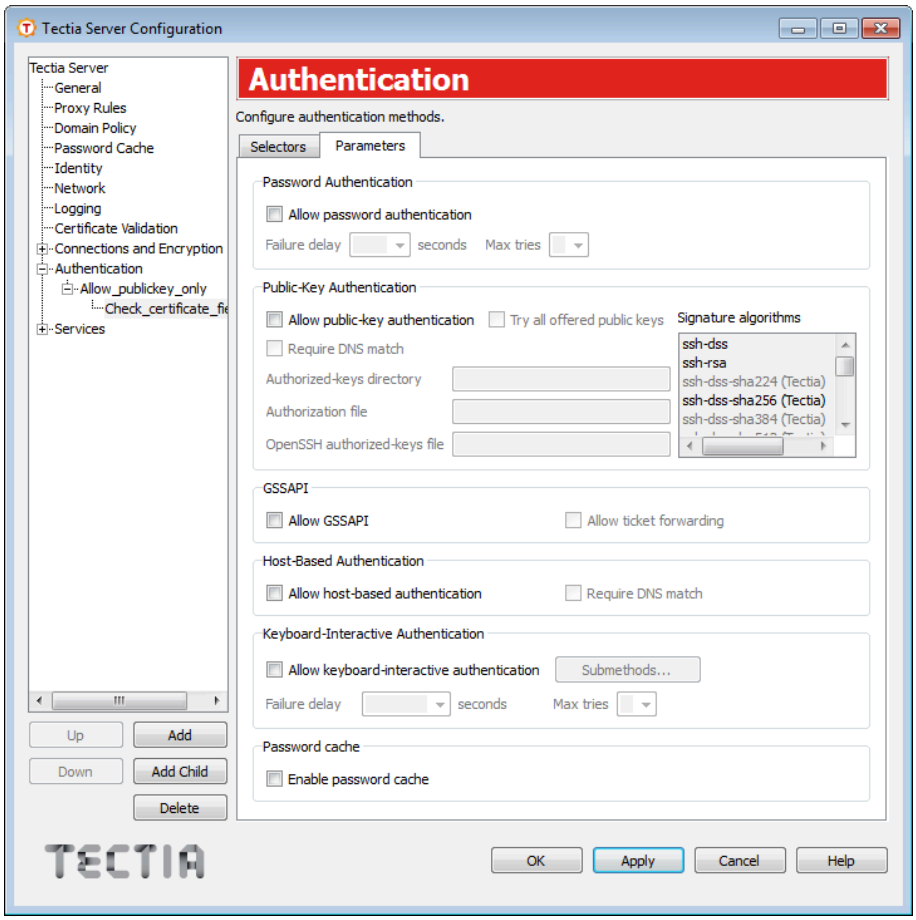


Figure 5.10. Unselecting authentication methods

14. Click **Apply** to save your changes.

You need to configure user authentication with certificates in Tectia Client also. For more information, see *Tectia Client User Manual*.

For more information about the authentication settings, see Section 4.1.12.

Troubleshooting User Authentication with Certificates

You can troubleshoot problems in user authentication with certificates by taking the following steps:

- Check that the server authentication phase is successful. When using x509v3 certificates, server authentication issues can sometimes stop client connections in the very beginning. Information about the server authentication issues must be checked from the client-side logs.
- Check the Windows Event Log. Tectia Server's log messages are stored into the Application sublog.

- If the logs do not show a clear reason for the user authentication problem, start Tectia Server in troubleshooting mode. Inspect the debugging messages using the **View Troubleshooting Log** tool in Tectia Server Configuration GUI (see Section 9.1.4).

5.8 Host-Based User Authentication

Host-based authentication uses the public key of the client machine to authenticate a user to the remote server. Host-based authentication can be used with Tectia Client on Unix. The Tectia Server can be either an Unix or Windows server. Usually also Tectia Server is installed on the client machine.

Host-based authentication provides a non-interactive form of authentication, and is best used in scripts and automated processes, such as cron jobs. Host-based authentication can be used to automate backups and file transfers, or in other situations where a user will not be present to input authentication information.

Caution

The nature of any non-interactive login is inherently unsecured. Whenever authentication without user challenge is permitted, some level of risk must be assumed. If feasible, public-key authentication is preferred. Tectia Server provides host-based authentication as a form of non-interactive login that is more secure than the `.rhosts` method used by the Berkeley 'r' commands, but it cannot resolve the inherent lack of security of non-interactive logins.

This means that you should take aggressive measures to ensure that any client machine set up for host-based authentication is adequately secured, both by software and hardware, to prevent unauthorized logins to the server.

Host-based authentication can be enabled either by using conventional public keys or by using certificates.

Note

When FIPS mode is enabled in the Server configuration (for more information, see **cryptolib**), host-based authentication only works if the file `libcrypto.*` (the file extension varies between platforms) resides in `/opt/tectia/lib/shlib/`.

In the following instructions, `Server` is the remote host running Tectia Server to which you are trying to connect. `ServerUser` is the user name on `Server` that you are logging in as. `Client` is the host running Tectia Client. `ClientUser` is the user name on `Client` that should be allowed to log in to `Server` as `ServerUser`. With Tectia Client, `ClientUser` and `ServerUser` must be the same user names.

5.8.1 Using Conventional Public Keys

Client Configuration

To enable host-based authentication with conventional public keys on the client, do the following as `ClientUser`:

1. Generate a host key. If Tectia Server has been installed on the same machine, the host key pair `/etc/ssh2/hostkey` and `/etc/ssh2/hostkey.pub` has been generated during installation and you can skip this step. Otherwise, give the following command:

   ```
   # ssh-keygen-g3 -P -H hostkey
   ```

   Optionally, you can define a custom location or name for the host key in the `ssh-server-config.xml` file. If Tectia Server is not installed on the client host, you can create the configuration file manually and save it in the `/etc/ssh2` directory.

2. Add the following line in the `ssh-broker-config.xml` file:

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

   Also other authentication methods can be listed. Place the least interactive method first (this means usually the host-based method).

Server Configuration

Do the following as the server administrator:

1. Copy the client's `/etc/ssh2/hostkey.pub` file over to the server. Note that this requires root permissions on the server, and may require root permissions on the client as well.

   Tectia Server looks for the host keys to use for host-based authentication in the `/etc/ssh2/trusted_hosts` directory on Unix and in the "`<INSTALLDIR>\SSH Tectia Server\trusted_hosts`" directory on Windows.

   You have to name the client's public key as follows on the server:

   ```
   client.example.com.ssh-dss.pub
   ```

   In the example, `client.example.com` is the host name that the client is sending to the server. When the server receives the client's public key, it forms a path based on the host name and the key type (`ssh-dss`, `ssh-rsa`, `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, `ecdsa-sha2-nistp521`, or `ssh-ed25519`) and compares the received public key to the key on the disk. If the public key matches and the user's login name in the remote end matches the name the user is trying to log in on the server, the user is let in after the signature check.

2. To enable host-based authentication on the server, in the `ssh-server-config.xml` file, under the `authentication-methods` element, add an `auth-hostbased` element:

   ```
   <authentication-methods>
     <authentication action="allow">
       <auth-hostbased require-dns-match="no" />
       ...
     </authentication>
   </authentication-methods>
   ```

Also other authentication methods can be allowed.

To force an exact match between the host name that the client sends to the server and the client's reverse mapped DNS entry, set the `require-dns-match` attribute to `yes`.

In this case, make sure the `/etc/hosts` file has the fully qualified domain name listed before the short host name, for example:

```
123.123.123.123   client.example.com   client
```

Even if you are not using `/etc/hosts` as your primary resolver, you may need to add entries to it for the client and the server to allow them to resolve each other's fully qualified domain names (if they are not able to do so otherwise).

Notice that when exact DNS matching is set as required, host-based authentication through NAT (Network Address Translation) will not work.

Using the **Tectia Server Configuration** tool, host-based authentication can be configured on the **Authentication** page. See Section 4.1.12.

3. Run `ssh-server-ctl` to take the new configuration in use. See ssh-server-ctl(8).

   Click **Apply** to take the new settings in use.

To test that host-based authentication works, log in to `Client` as `ClientUser` and run the following command:

```
$ sshg3 ServerUser@server uptime
```

You should get back the results of uptime on the server.

5.8.2 Using Certificates

It is possible to use a certificate instead of the conventional public-key pair to authenticate the client host.

The endpoint identity check, where the server verifies that the certificate actually belongs to the client that is attempting host-based authentication, is performed according to the following rules:

1. One of the DNS subject alternative names in the client certificate must match the client's fully qualified domain name obtained by doing a reverse lookup on the client's IP address. The alternative names may have an asterisk (`*`) as the first component, in which case only the domain part is checked.

2. If the client's IP address cannot be reverse-mapped, the IP address is compared to the certificate's IP subject alternative names.

3. If the above checks do not produce a positive result, the certificate's subject name is checked. If it has a CN component that matches the client's reverse-mapped fully qualified domain name or IP address, the certificate is accepted.

Client Configuration

To enable host-based authentication with certificates on `Client`, make the following settings in the Connection Broker configuration on the client side:

1. Add the following line in the `ssh-broker-config.xml` file:

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

   Also other authentication methods can be listed. Place the least interactive method first (this means usually the host-based method).

2. Enroll a certificate for `Client`. See Section 5.7 for more information.

   The certificate must contain a `dns` extension which contains the fully qualified domain name (FQDN) of `Client`.

Note

The private key associated with the certificate needs to be stored with an empty passphrase.

3. Define the private key and certificate in `ssh-server-config.xml` on `Client`:

```
<params>
  <hostkey>
    <private file="/etc/ssh2/hostcert" />
    <x509-certificate file="/etc/ssh2/hostcert.crt" />
  </hostkey>
...
</params>
```

   If Tectia Server is not installed on `Client`, create the configuration file manually and save it in the `/etc/ssh2` directory.

Server Configuration

Do the following as the server administrator:

1. Specify the CA certificate in the `ssh-server-config.xml` file:

```
<cert-validation>
  <ca-certificate name="exa-ca1" file="/etc/ssh2/exa-ca1.crt" />
  ...
</cert-validation>
```

2. In the `ssh-server-config.xml` file, under the `authentication-methods` element, add an `auth-hostbased` element and define the selectors. For example:

```
<authentication-methods>
  <authentication name="hostbased-block">
    <auth-hostbased require-dns-match="no" />
```

```
    <authentication action="allow" name="hostbased-cert-allow">
      <selector>
         <host-certificate field="ca-list" pattern="exa-ca1" />
         <host-certificate field="issuer-name" pattern="C=FI, O=SSH, CN=*" />
         <host-certificate field="serial-number" pattern="123456" />
         <host-certificate field="altname-ip" pattern="10.0.1.10" />
         <host-certificate field="altname-fqdn" pattern="*.ssh.com" />
       </selector>
    </authentication>
    <authentication action="deny" />
  </authentication>
</authentication-methods>
```

The host-based authentication with certificates can be restricted using the following `field` attributes in the `selector`:

- `ca-list`: The pattern is a comma-separated list of CA names. The names that are defined in the `ca-certificate` element are used.
- `issuer-name`: The pattern is the required certificate issuer name in LDAP DN (distinguished name) string format. The issuer name may contain glob patterns ('*' and '?') but only in the component values, not names. For example, `"C=FI, O=SSH, CN=*"` is a legal pattern, but `"C=FI, *=SSH, CN=TestCA"` is not).
- `subject-name`: The pattern is the required subject name in LDAP DN (distinguished name) string format. Matching is done in similar manner as with the issuer name described above.
- `serial-number`: The pattern is the required serial number of the certificate. A combination of issuer name and serial number can be used to uniquely identify a certificate.
- `altname-email`: The pattern is the e-mail address that must be present in the certificate as a subject alternative name.
- `altname-upn`: The pattern is the principal name that must be present in the certificate as a subject alternative name.
- `altname-ip`: The pattern is the IP address that must be present in the certificate as a subject alternative name. Also a range of addresses can be given (for example, `10.1.0.11-10.1.0.61` or `10.1.0.0/8`).
- `altname-fqdn`: The pattern is a list of fully qualified domain names (FQDN) that may contain glob patterns ('*' and '?'). One of the listed domain names must match with a subject alternative name of type FQDN in the certificate.

In addition to matching to the selectors, the certificate must pass the endpoint identity check, described in detail in Section 5.8.2.

Using the **Tectia Server Configuration** tool, host-based authentication can be configured on the **Authentication** page. See Section 4.1.12.

3. Run `ssh-server-ctl` to take the new configuration in use. See ssh-server-ctl(8).

Click **Apply** to take the new settings in use.

5.9 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.

Currently, the following keyboard-interactive submethods are supported:

- password
- PAM (Unix only, see note below)
- RSA SecurID
- RADIUS
- LAM (AIX only)

Methods that require passing some binary information, such as public-key authentication, cannot be used as submethods of keyboard-interactive. But public-key authentication, for example, can be used as an additional method alongside keyboard-interactive authentication.

Note

PAM has support also for binary messages and client-side agents, but those cannot be supported with keyboard-interactive.

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. The server can be configured to allow, for example, the two optional submethods SecurID and password, and then the user can skip SecurID by pressing **Enter** when the server asks for a SecurID. The user will then be prompted for a password.

Using the **Tectia Server Configuration** tool, keyboard-interactive authentication can be configured on the **Authentication** page. See Section 4.1.12.

5.9.1 Password Submethod

Password authentication can also be used over keyboard-interactive.

The following example shows settings for allowing keyboard-interactive authentication using the password submethod in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
```

```
    <auth-keyboard-interactive max-tries="3" failure-delay="2">
      <submethod-password />
    </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

5.9.2 Pluggable Authentication Module (PAM) Submethod

Pluggable Authentication Module is an authentication framework used in Unix systems. In Tectia, support for PAM is enabled as a submethod of keyboard-interactive authentication.

When PAM is used, Tectia Server transfers the control of authentication to the PAM library, which will then load the modules specified in the PAM configuration file. Finally, the PAM library tells Tectia Server whether or not the authentication was successful. Tectia Server is not aware of the details of the actual authentication method employed by PAM, only the final result is of interest.

The PAM authentication can be enabled by creating a PAM configuration for the service `ssh-server-g3`. For information on how to do PAM session and account management irrespective of the authentication methods used, see the configuration element description for **pluggable-authentication-modules** .

In addition, you can define separate authentication blocks with specific PAM settings (in element `<submethod-pam>`) which will override the PAM defaults for that particular authentication block.

It is possible to configure the user session management and the authentication to use different services for PAM authentication. This is done by defining different services in the `service-name` attribute in the `pluggable-authentication-modules` element and in the `submethod-pam` element.

Tectia Server expects to find the PAM libraries in the default paths of the supported operating systems. You need to define the PAM libraries in the server configuration only if they are used from non-default locations.

The following configuration example shows the PAM authentication related settings in the `ssh-server-config.xml` file.

```
<authentication-methods>
  <authentication action="allow">
    <auth-keyboard-interactive>
      <submethod-pam
         service-name="sshd2"
         dll-path="path-to-pam-dll" />
    </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

Note

SSH Communications Security does not provide technical support on how to configure PAM. Our support only covers Tectia applications.

PAM Examples

The following are examples of the PAM configurations on different platforms.

Please note that these are just examples and need to be modified according to the actual server configuration.

PAM on Red Hat Linux

On Red Hat Linux 5, add PAM configuration file `/etc/pam.d/ssh-server-g3` with contents:

```
auth       include      system-auth
account    required     pam_nologin.so
account    include      system-auth
password   include      system-auth
session    optional     pam_keyinit.so force revoke
session    include      system-auth
session    required     pam_loginuid.so
```

When the PAM library is used from the default path, the PAM definitions in the Tectia Server configuration file `ssh-server-config.xml` can be simply as follows:

```
<authentication-methods>
  <authentication action="allow">
    <auth-keyboard-interactive >
        <submethod-pam />
   </auth-keyboard-interactive >
  ...
  </authentication>
</authentication-methods>
```

PAM on SUSE Linux

On SUSE Linux Enterprise Server 10 (both 32- and 64-bit versions) the default configuration settings are suitable for most PAM authentications. You can add file `/etc/pam.d/ssh-server-g3` with contents:

```
auth     include        common-auth
auth     required       pam_nologin.so
account  include        common-account
password include        common-password
session  include        common-session
```

The following example configuration in `ssh-server-config.xml` enables PAM session and account management with the service `sshd2` (instead of the default `ssh-server-g3`). The authentication submethod PAM is configured to use service `ssh-server-g3`. The PAM library is used from the default path.

```
<params>
  <pluggable-authentication-modules
            service-name="sshd2"
            pam-calls-with-commands="yes" />
</params>
         <!-- ... -->
<authentication-methods>
```

```
  <authentication action="allow" name="allow-default">
    <auth-keyboard-interactive>
      <submethod-pam service-name="ssh-server-g3"/>
    </auth-keyboard-interactive>
  </authentication>
</authentication-methods>
```

PAM on AIX

On AIX, the PAM library is able to recognize whether the calling application is 32- or 64-bit and then substitute the correct path to load modules if full path has not been specified in the `/etc/pam.conf` file. If the `pam.conf` file has the following specified for **ssh-server-g3**, it should work with both Tectia Server versions 6.1 and 6.2:

```
ssh-server-g3 auth     required pam_aix
ssh-server-g3 account  required pam_aix
ssh-server-g3 password required pam_aix
ssh-server-g3 session  required pam_aix
```

Note

If PAM authentication is in use and you are updgrading to Tectia Server 6.6, which is a 64-bit version, note that if the full path is set in the `pam.conf`, it points to 32-bit PAM modules for Tectia Server.

PAM on Oracle Solaris

On Solaris versions 10 and 11, add the `/etc/pam.conf` entry with contents:

```
ssh-server-g3   auth requisite          pam_authtok_get.so.1
ssh-server-g3   auth required           pam_dhkeys.so.1
ssh-server-g3   auth required           pam_unix_cred.so.1
ssh-server-g3   auth required           pam_unix_auth.so.1
ssh-server-g3   account requisite       pam_roles.so.1
ssh-server-g3   account required        pam_unix_account.so.1
ssh-server-g3   session required        pam_unix_session.so.1
ssh-server-g3   password required       pam_dhkeys.so.1
ssh-server-g3   password requisite      pam_authtok_get.so.1
ssh-server-g3   password requisite      pam_authtok_check.so.1
ssh-server-g3   password required       pam_authtok_store.so.1
```

If the PAM library is used from a path different than the operating system default, the path must be specified in the Tectia Server configuration file `ssh-server-config.xml` both in the `pluggable-authentication-modules` and in the `submethod-pam` element with the `dll-path` attribute. For example:

```
<params>
  <pluggable-authentication-modules
            dll-path="path-to-pam-dll"
            pam-calls-with-commands="yes" />
</params>

<authentication-methods>
```

```
  <authentication action="allow" name="allow-default">
    <auth-keyboard-interactive>
      <submethod-pam dll-path="path-to-pam-dll" />
    </auth-keyboard-interactive>
  </authentication>
</authentication-methods>
```

Note

On Solaris, the account lockout setting LOCK_AFTER_RETRIES in `/etc/security/policy.conf` only applies if keyboard interactive authentication is used with PAM. Other types of authentication methods do not increment the retries count.

PAM Used with LDAP on Red Hat Linux

The following is an example on how to configure PAM to use LDAP authentication on a Red Hat Linux machine. Before trying this setup, verify that PAM works for local accounts. Modify the example settings according to your LDAP server configuration.

In file `/etc/pam.d/ssh-server-g3`, add the following settings:

```
auth     required /lib/security/pam_ldap.so
account  required /lib/security/pam_ldap.so
password required /lib/security/pam_ldap.so
session  required /lib/security/pam_ldap.so
```

In file `/etc/nsswitch.conf`, add the following settings:

```
passwd: files ldap
shadow: files ldap
group: files ldap
```

In file `/etc/ldap.conf`, add the following settings:

```
host ldapserver.company.com
base dc=company,dc=com
ldap_version 3
port 389
scope one
pam_min_uid 10000
pam_max_uid 20000
nss_base_passwd ou=accounts,dc=company,dc=com?one
nss_base_shadow ou=accounts,dc=company,dc=com?one
nss_base_group ou=groups,dc=company,dc=com?one
ssl no
pam_password md5
```

5.9.3 RSA SecurID Submethod

RSA SecurID is a widely-used two-factor authentication method based on the use of SecurID Authenticator tokens. In Tectia, support for RSA SecurID is enabled as a submethod of keyboard-interactive authentication.

The prerequisite for enabling SecurID support in Tectia Server is that *RSA Authentication Agent* software (previously *RSA ACE/Agent*) is installed on the server host.

Note

To enable SecurID support in Tectia Server on a 64-bit Windows server host, do the following:

1. Install the 32-bit *RSA Authentication Agent* on a 32-bit Windows system.
2. Copy the `aceclnt.dll` and `sdmsg.dll` files from the `C:\Program Files\Common Files\RSA Shared\Auth Data` directory and place the files on the 64-bit Windows server host in the `C:\Program Files (x86)\SSH Communications Security\SSH Tectia\SSH Tectia Server` directory with the `sdconf.rec` file from the *RSA Authentication Manager*.
3. Add `C:\Program Files (x86)\SSH Communications Security\SSH Tectia\SSH Tectia Server` in the System Path of the Windows Environment Variables.

When RSA SecurID is used, Tectia Server queries the user for the token's numerical code and passes the code to RSA Authentication Agent for verification. RSA Authentication Agent then returns the success or failure of the authentication to Tectia Server.

RSA SecurID authentication provides two different authentication agents/:

- RSA Authentication Agent for PAM (versions 5.3.4 and 6.0.0)
- RSA Authentication Agent for UNIX (version 5.2)

The Tectia Server configuration needs different settings depending on which RSA Authentication Agent is used. For configuration examples, see the section called “Configuring RSA Authentication Agent for Unix” and the section called “Configuring RSA Authentication Agent for PAM”.

To use SecurID authentication, you should be familiar with the operation of *RSA Authentication Manager* (previously *RSA ACE/Server*).

Configuring RSA Authentication Agent for Unix

For the SecurID authentication to work with Tectia Server on Unix, the RSA Authentication Agent `libaceclnt.so` library has to be available in the `/usr/lib` directory (alternatively `/user/ace/lib` or `/opt/ace/lib`).

The following example shows the settings required in the `ssh-server-config.xml` file for keyboard-interactive authentication using the SecurID submethod:

```
<authentication-methods>
  <authentication action="allow">
    <auth-keyboard-interactive max-tries="3" failure-delay="2">
      <submethod-securid />
    </auth-keyboard-interactive>
  ...
  </authentication>
```

```
</authentication-methods>
```

Giving the `dll-path` attribute is not required. Tectia Server locates the libraries automatically.

Configuring RSA Authentication Agent for PAM

When you want to use keyboard-interactive authentication using the RSA Authentication Agent for PAM, make the following settings in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
    <auth-keyboard-interactive>
      <submethod-pam />
    </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

In addition, create a symlink for `libpam` as follows:

```
ln -s /lib/libpam.so.0 /lib/libpam.so
```

Create the `/etc/pam.d/ssh-server-g3` file containing:

```
auth     required  /lib/security/pam_securid.so
acccount required  /lib/security/pam_pwdb.so
session  required  /lib/security/pam_pwdb.so
```

For more information, see the separate *RSA SecurID Ready Implementation Guide* for Tectia, available from the RSA web site (http://www.rsasecured.com/).

Note

SSH Communications Security does not provide technical support on how to configure *RSA Authentication Manager* (*RSA ACE/Server*). Our support only covers Tectia applications.

5.9.4 RADIUS Submethod

RADIUS (Remote Authentication Dial-In User Service) is a protocol for checking a user's authentication and authorization information from a remote server. It was originally intended for authenticating dial-in users, but is also suitable for use with Secure Shell. In Tectia, RADIUS is implemented as a submethod of keyboard-interactive authentication.

When using RADIUS authentication, Tectia Server first asks the user's password and then sends it along with the user name to the RADIUS server (PAP authentication). Multiple RADIUS servers can be configured, and these will be queried in turn in case some of them are unreachable.

The supported RADIUS servers are Microsoft IAS (Internet Authentication Service) and FreeRADIUS.

The following example shows settings for keyboard-interactive authentication using the RADIUS submethod in the `ssh-server-config.xml` file:

```
<authentication-methods>
  <authentication action="allow">
    <auth-keyboard-interactive max-tries="3" failure-delay="2">
      <submethod-radius>
        <radius-server address="10.1.61.128"
                       port="1812"
                       client-nas-identifier="nasid">
          <radius-shared-secret file="&configdir;/radius-secret-file" />
        </radius-server>
      </submethod-radius>
   </auth-keyboard-interactive>
  ...
  </authentication>
</authentication-methods>
```

Using the **Tectia Server Configuration** tool, keyboard-interactive authentication can be configured on the **Authentication** page. See Section 4.1.12.

Notice that enforcing password changing does not work with RADIUS.

A common cause of problems in RADIUS authentication is that the shared secret is corrupted. For example, extra newline characters or spaces in the shared secret file can cause the authentication to fail. Make sure the same shared secret is configured on Tectia Server and the network access server (NAS).

Note

SSH Communications Security does not provide technical support on how to configure RADIUS. Our support only covers Tectia applications.

For information on configuring FreeRADIUS, see for example, http://www.freeradius.org/. For information on configuring Microsoft IAS, see its documentation.

Special Considerations on Windows

When using RADIUS authentication to log on to a Windows server that belongs to a domain, you have to give the user name prefixed with the machine name, for example `MACHINE\user` (instead of `user`). This is because RADIUS authentication uses local accounts, and Tectia Server that is installed on a Windows domain machine assumes that user accounts given without a prefix are domain accounts.

If Tectia Server is installed on a stand-alone machine, you can use both notations with RADIUS authentication (`MACHINE\user` and `user`).

For more information about user accounts on Windows, see Section 5.5.3.

5.9.5 LAM Submethod on AIX

AIX systems use the Loadable Authentication Module (LAM) as their default subsystem for providing the identification and authentication facilities. In Tectia Server, support for LAM can be enabled as a submethod of keyboard-interactive authentication which uses an AIX-LAM plugin.

When LAM is used, Tectia Server transfers the control of authentication to the LAM library, and expects in return information on whether the user authentication was successful or not. Tectia Server does not need to be aware of the details of the actual authentication methods employed by LAM, it only reacts to the returned authentication success result.

The AIX-LAM plugin can also be enabled to request password changing in case the user password has expired. To enable LAM on Tectia Server running on AIX, and to allow also changing of the password, use the following settings in configuration file `ssh-server-config.xml`:

```
<authentication-methods>
      <authentication name="authentication">
        <auth-keyboard-interactive >
           <submethod-aix-lam enable-password-change="yes" />
        </auth-keyboard-interactive >
      </authentication>
</authentication-methods>
```

5.10 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 2003 (or later) 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 server event log.

To enable GSSAPI authentication on the server, the `authentication-methods` element of the `ssh-server-config.xml` file must contain an `auth-gssapi` element. For example:

```
<authentication-methods>
  <authentication action="allow">
    <auth-gssapi dll-path="path-to-gssapi-dll" />
  ...
  </authentication>
</authentication-methods>
```

Also other authentication methods can be allowed.

Using the **Tectia Server Configuration** tool, GSSAPI authentication can be configured on the **Authentication** page. See Section 4.1.12. On Windows, the `dll-path` attribute is ignored. Tectia Server locates the correct DLL automatically.

Note

SSH Communications Security does not provide technical support on how to configure Kerberos. Our support only covers Tectia applications.

5.11 Supplementing Authentication with an External Application

Tectia Server allows using an external application to supplement authentication. This also makes it possible to use information stored in an external database to allow access for specific users.

The external application, which may be written in any programming language suitable for the task, talks to Tectia Server using the Tectia Mapper Protocol. (For more information on the protocol, see Appendix E.)

The path to the external application is defined in the `ssh-server-config.xml` file within an `authentication` block, using the **`mapper`** element's `command` attribute.

Caution

The external application will be launched under administrator (root) privileges.

Tectia Server sends data from its **blackboard** to the external application. For a detailed description of the data that the server sends, see **mapper** in ssh-server-config(5). The data that the external application sends back to Tectia Server will be stored in the server's blackboard.

For the authentication to succeed, the external application must return "`success`" and an exit status 0. For more information on the parameters allowed by Tectia Mapper Protocol, see Section E.1.

Sample scripts written in Python are provided in `/etc/ssh2/samples` on Unix and `<INSTALLDIR>\SSH Tectia AUX\samples` on Windows.

5.11.1 Example with Certificate Authentication

This example presents a typical use case for user mapping: matching a certificate and a user. Selectors are usually used for this purpose, but if you have a database that contains information about users and certificates, you can use an external application defined in the `mapper` element to query the database. Based on the query result, users can be allowed/denied access to the server.

In this example the user is allowed to log in only if the regular expression provided in the `certificate` element matches, that is if `CN` in the certificate's subject name consists of three words separated by periods (.) and a set of digits in the end (for example "`Smith.John.James.1234`").

```
<authentication-methods login-grace-time="600">
  <authentication action="allow">
    <auth-publickey />
    <authentication name="authentication3" action="allow">
      <selector>
```

```
        <certificate field="subject-name"
         regexp="C=FI, O=SSH, CN=\\w\+\\.\\w\+\\.\\w\+?\\.?\\w\+?\\.\\d\+" />
      </selector>
      <mapper command="/path/to/python /path/to/script1.py"/>
    </authentication>
  </authentication>
</authentication-methods>
```

5.11.2 Example with Password Authentication

In this example the user is requested to provide a password, and the external application (`/path/to/script2.py`) is additionally used to check whether the user should be allowed access.

```
<authentication-methods login-grace-time="600">
  <authentication action="allow">
    <auth-password />
    <mapper command="/path/to/python /path/to/script2.py"/>
  </authentication>
</authentication-methods>
```

5.12 Configuring User Authentication Chains

The user authentication configuration in Tectia Server 6.x has been significantly upgraded as compared to 4.x and earlier versions. It is much more versatile but also more complex. This section includes several examples of user authentication configuration to give administrators an insight into what the new system can do.

5.12.1 Basic Example

Figure 5.11 shows the simplest possible authentication chain example. It contains one authentication block, which contains one method definition.

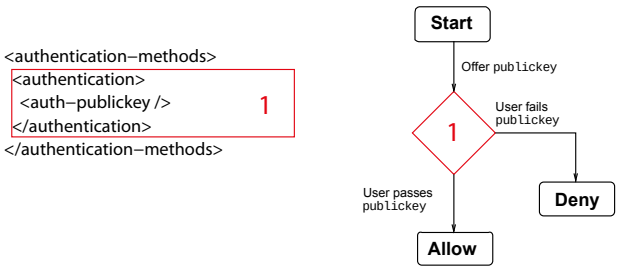


Figure 5.11. Basic authentication example

When the server starts the user authentication exchange with the client, it enters the authentication block marked with `1`, gathers the list of methods defined in it and sends that list to the client to inform it of the acceptable authentication methods. In this example, only one method is allowed.

Whenever the user passes any one of the methods, it is considered to have satisfied the authentication block. After that, the server either proceeds to a nested authentication block, or if there are no further blocks to enter, marks the user as authenticated. Once the server has entered an authentication block, it will never exit it – the processing continues only inside the block.

5.12.2 Example with Selectors

The example in Figure 5.12 demonstrates the use of multiple authentication blocks with selectors. Selectors match against information gathered during the connection attempt and may be used to control the user authentication process.

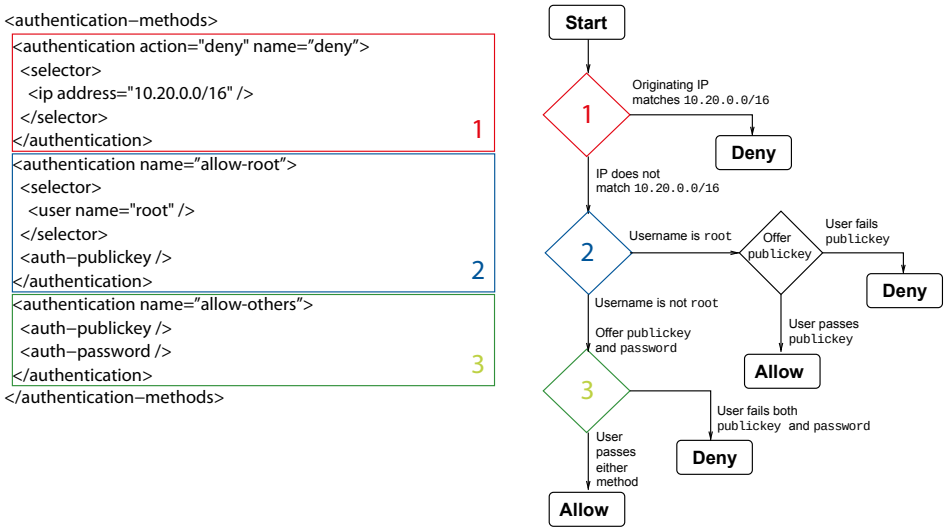


Figure 5.12. Authentication example with selectors

In this example, there are three authentication blocks on the same level. The authentication processing enters the first block that has a matching selector. A block without selectors always matches, so such block must always be the last in order.

The flowchart in the figure demonstrates the matching process. The server considers each authentication block in turn and either moves on or enters the block. Once a block has been entered, the processing is confined to that block only (and possible nested blocks inside it).

Each authentication block should either be a terminal block (contain an action definition, "`deny`" or "`allow`") or continue the processing by having either one or more authentication methods or nested authentication blocks. (A block with no methods, nested blocks, or action definition, is considered to be an `allow` block.)

5.12.3 Authentication Chain Example

The previous example showed how the server may be configured to select one authentication block from a list of multiple blocks using selectors. This allows the administrator to select a list of allowed authentication methods according to user name, originating IP address, and various other attributes.

However, this does not allow the administrator to require more than one authentication method. The way to do this is by creating a chain of nested authentication blocks.

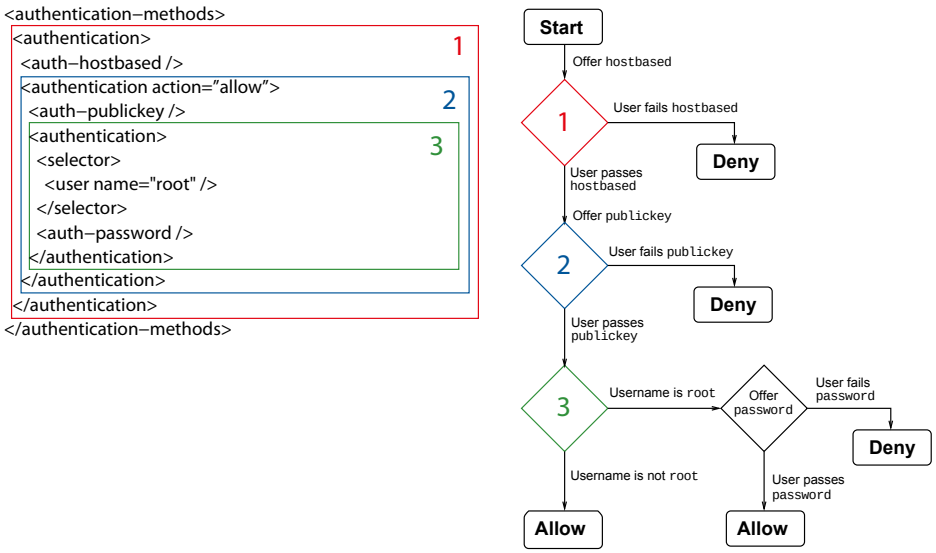


Figure 5.13. Authentication chain example with nested authentication methods

In the example shown in Figure 5.13, the top-level block (marked with `1`) contains one method definition, `hostbased`. When starting the authentication exchange with the user, the server sends only that method as allowed to the client. If the user fails that method, the whole authentication fails. If the user passes the method, the server looks for a nested block (marked with `2`), forms a list of methods defined in that block and sends that list to the client. In this example, the nested block contains the `publickey` method.

After the user has passed public-key authentication, the server looks for further blocks for a continuation. In this example, there is one nested block at the innermost level (`3`). The block is selected only if the user is trying to log in as `root`. In that case, one more authentication method is required.

If the user does not match to a nested authentication block, the action of the parent authentication element is used (in this example, `allow` in step 2). Users logging in with other user names than `root` will be allowed in after having passed both `hostbased` and `publickey` earlier.

In step 2, the `allow` action is shown for clarity. The `allow` attribute can also be omitted from the configuration as it is the default action.

This example contains only one method at each level and results in one method being required at a time. It would also be possible to have multiple method definitions at each level, in which case passing any one of the methods would allow the authentication to proceed to the next level.

5.12.4 Example of Using the Deny Action

The example in Figure 5.14 illustrates using the deny action with nested authentication methods.

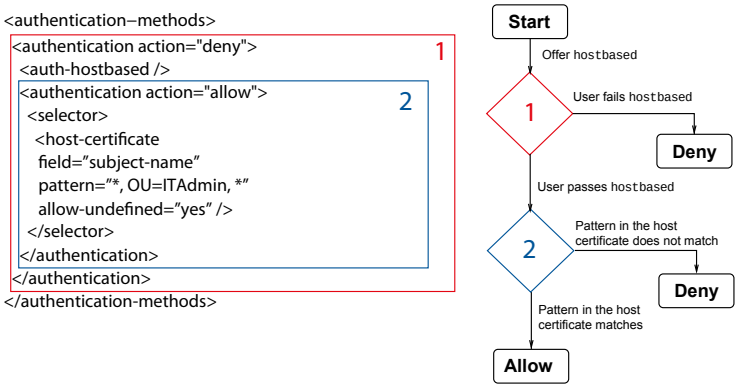


Figure 5.14. Using the deny action with nested authentication methods

When the user authentication processing starts, the user is first directed to a block with one authentication method, `hostbased`. If the user fails host-based authentication, the processing immediately ends in failure.

If the user passes host-based authentication, the authentication state is still "deny" and the processing continues with a nested authentication block. This block matches if a host certificate with valid fields was used to pass the host-based authentication. In this case, no further authentication methods are required and the authentication ends in success.

However, if a matching certificate does not exist, the deny action of the parent authentication block is used and the authentication ends in failure.

The `allow-undefined` attribute is included in the host certificate selector and set to "`yes`". If it is omitted (or set to "`no`"), and the user tries to authenticate with a normal host public key, the selector matching will end in error because the host certificate data is not available to the server. In this example, the end result would be the same from the user's point of view (login is denied), but the server logs this as an error condition instead of an authentication failure. For more information, see the section called “Selectors and Undefined Data”.

5.13 Forwarding User Authentication

Tectia client/server solution supports user authentication forwarding with public-key and certificate authentication methods. Secure Shell connections and public-key authentication data can be 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 host, and authentication passphrases or private keys never go over the network.

For more information, see Section 8.5.

5.13.1 Forwarding User Authentication to a Kerberos Realm

Tectia client/server solution supports authenticating to a Kerberos realm with authentication agent forwarding with the private keys stored on the local host. This makes it possible to log in to a Kerberos realm from a second client/server host during a Secure Shell session.

For example, when a Tectia Client user uses certificates (or a smartcard token) to connect via Tectia Server to other remote servers (running any Secure Shell servers), the local Connection Broker can act as a key store and provide the user's keys to a third-party application such as MIT Kerberos for authentication.

When Tectia Server and **sshg3** are used on the second host, the key provider socket is set up by default, as long as authentication agent forwarding is allowed by both Connection Broker and Tectia Server.

Example of the required configuration in the `pkcs11` module in the `krb5.conf` of MIT Kerberos on the second host:

```
[realms]
 DOMAIN.COM = {
  kdc = ad.domain.com:88
  kpasswd_server = ad.domain.com:464
  pkinit_kdc_hostname = ad.domain.com
  pkinit_identities = PKCS11:/opt/tectia/lib/sshack.so
  pkinit_anchors = FILE:/etc/krb5/ca.crt
  pkinit_win2k = true
  pkinit_eku_checking = kpServerAuth
  pkinit_cert_match = <SAN>.*@DOMAIN.COM
  forwardable = true
  forward = true
 }
```

Connection Broker configuration on the second host should include:

```
<default-settings>
  <authentication-methods>
    <auth-gssapi />
    <auth-publickey />
    <auth-keyboard-interactive />
    <auth-password />
  </authentication-methods>
  <forwards>
    <forward type="agent" state="on" />
  </forwards>
</default-settings >
```

`/opt/tectia/lib/sshack.so` implements a set of PKCS #11 functionality to support signing, which is not restricted to usage in Kerberos.

If the connection is made via other tools than Tectia Client, the environment variable `SSH_AA_SOCK` on the local host needs to be configured with the path to the Connection Broker agent socket (by default, `/tmp/ssh-<user>/ssh-broker-aa`).

5.14 Reporting User Login Failures

This section explains how Tectia Server reports user login failures to the operating system. Notice that this is different from recording login events to the audit system. The operating system can use the login failure reports to block user accounts, for instance.

When password authentication or keyboard-interactive authentication with passwords is used, Tectia Server will report every failing password login attempt to the operating system.

With all other authentication methods on AIX and HP-UX, Tectia Server will report one login failure when the connection has been disconnected, if no login failures have been reported previously. On other Unix platforms, Tectia Server does not report any login failures in this case.

When any third-party authentication methods are used, such as keyboard-interactive authentication with PAM, the used method may report the login failures to the operating system independently.

For example:

- If public key and PAM are used as the authentication methods, and all user login attempts fail, Tectia Server will report only one login failure.
- If public key, PAM and password (with 5 attempts allowed) are used as the authentication methods, and all attempts fail, Tectia Server will report 5 login failures.

5.15 User Name Handling on Windows

On Windows, the user name is generally handled similarly irrespective of which user authentication method you use: password, public keys, certificates, keyboard-interactive, or GSSAPI.

The user can define a prefix together with the user name. The prefix indicates whether it is a local or a domain user name. If the user provides the prefix, Tectia Server always handles the user name according to that.

If no prefix is provided with the user name, Tectia Server by default treats logon user names as user accounts of the default domain in case the computer belongs to a domain, but if no match is found, the user will be treated as a local user. This default policy can be overriden by defining a domain policy in the Tectia Server configuration. If the user defines a prefix in the user name, that will override both the default policy and the domain policy.

The domain policy can be defined in the Tectia Server configuration on the **Domain Policy** page of the **Tectia Server Configuration** tool (see Section 4.1.4) or with the `domain-policy` element in the XML configuration file (see description of **domain-policy**).

Table 5.2. Principles of user name handling on Windows hosts in a domain

Situation	How the Tectia Server treats user names
Prefix defined:	According to the prefix
No prefix defined:	
Domain policy defined in Tectia Server configuration →	According to the domain policy
No domain policy defined →	1. Domain user name tried first

Situation	How the Tectia Server treats user names
	2. Local user name tried only if domain name not found

Normally when logging on to a server, you specify the target computer and optionally your user name, for example:

```
$ sshg3 win-server
```

OR

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

In case the user does not specify the user name or a prefix for it indicating whether it is a local or a domain user name, and if the Windows server belongs to a domain for which no domain policy has been defined, the user name is assumed to be a domain user name and the name of the server's default domain is added as the prefix when checking the existence of a user account:

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

If no prefix is specified by the user, and if no matching domain user name is found, the user will be treated as a local user and the local computer name is automatically added as the prefix when checking the existence of a user account:

```
win-server\user@win-server
```

Tip

If you want to make sure that the local user name is used to log on to a Windows domain machine, you have to explicitly indicate that you are using a local account. You can either specify the local machine name as the domain part of the user name: `win-server\user@win-server`; or you can use shorthand notation **/** or **** as follows: `/user@win-server`.

The shorthand notation with **/** or **** is applicable in case the user does not know the host names but connects using the IP address. The shorthand notation is also a quick way of avoiding repeating long host names.

5.16 Requirements for Trusted Domain Authentication on Windows

This section describes the requirements for allowing trusted domain authentication in Windows domains. These requirements apply to any passwordless authentication method when Tectia Server is located in another Windows domain than the client users accessing Tectia Server and services it offers. The client users may be located in a network domain that is external to a corporate network providing a service that is secured with Tectia Server. These requirements apply to Windows domain controllers only.

Domain controllers

Windows Server 2008 or a newer version is required.

Trust path between domains

A bidirectional trust path between Windows domains is required when the client and the service are in different domains. Otherwise Kerberos extensions from Microsoft called Service-for-User (S4U) do not work. If bidirectional trust cannot be used, you can set up a one-way trust relationship using the **Tectia Server Configuration**, tool **Domain Policy** page (see Section 4.1.4) or with the `windows-domain` element in the XML configuration file.

Functional level of domains

The functional level of domain controllers should be `Native` `Win2003` in order for the Kerberos extensions to work properly.

You can raise the domain functional level by logging into the primary domain controller with administrator credentials. Locate the Active Directory Users and Computers and in the console tree, right-click the domain node whose functional level you want to raise.

DNS suffixes

DNS suffixes must be configured properly so that the trusted domains can see each other and can retrieve information about users.

On the DNS server, by clicking the Advanced button in a connection's Internet Protocol (IP) Properties dialog box, you can open the connection's Advanced TCP/IP Settings dialog box. On the DNS tab of this dialog box, you can create DNS suffixes to be used by the connection.

5.17 Accessing Resources on Windows Network from Logon Sessions Created by Tectia Server

When access to resources on Windows network from Tectia Server's logon session is needed:

- Use the password authentication method to achieve the most benefits. If you use two-factor authentication with Radius or SecurID, use it together with password authentication.

  It is possible to log into a local or domain account. The remote computer on Windows network can be in the domain, but that is not required.

- When computers on Windows network of Tectia Server are in the same domain with the SSH server, you can try using also other authentication methods that do not involve native Windows account credentials. Tectia Server uses Microsoft S4U2Self (Service-for-User-to-Self) method to obtain the user's access token. This method must be used together with constraint delegation of authentication configured on your domain controller for specific resources that are going to be accessed.

Example: John Brown wants to access some files on several Windows file servers in his company's private network. The only access point to his company's network is via Tectia Server on a Windows server that serves as a gateway to the internal network. File share called `share1` on `server1` is mapped to drive `N:`

in his user account on the SSH server host. John also wants to access `share2` from `server2` via UNC path. See Figure 5.15:

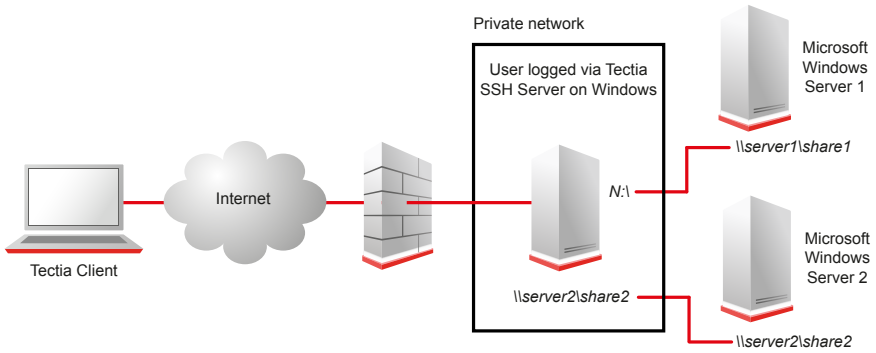


Figure 5.15. Accessing files on several Windows servers via Tectia Server

It is important to remember that a user who logs on to a Windows server via Tectia Server (or any other means) is authorized to use only that specific host. If the user wants to access other computers in the local network of the server host, the authentication to those computers must be performed first. Microsoft Windows operating systems already provide the means to minimize the number of interactive password prompts when working in network environments. Examples of this are credential cache implemented in credential manager of specific network provider and Kerberos Ticket Granting Ticket.

Tectia Server does not implement the network provider itself, neither it implements the Kerberos authentication module. Therefore it does not implement any credential management integration into Windows OS as such. As a consequence, if a user wants to access network resources from Tectia Server logon session, it is important to log on to SSH server using the right authentication method. Currently, the only authentication methods that can benefit from credential management of Windows network providers are the methods that use native Windows credentials (the user name and password of a Windows local or domain account). These are the password authentication method (see Section 5.5) and the password submethod of keyboard-interactive authentication method (see Section 5.9.1).

Whenever a user who logged on to the system using a user name and password tries to access network resources, the operating system will first try to use cached credentials provided during logon. If the user has access rights to such network resources, the user will be authenticated to a remote machine and access will be granted. If the user is not allowed to access the resources, then the user may be prompted to provide alternative credentials. If interaction is not possible, the attempt will fail.

The following happens when a user logs on to Tectia Server using an authentication method that does not use the user's Windows user name and password:

1. Tectia Server logs the user on to the system without providing the user's password. Currently, Tectia Server first tries to obtain the user's access token using S4U2Self - Microsoft extension of Kerberos protocol. If this fails (it is only supported on Windows server platforms), then Tectia Server's own authentication package SSHDAP is used.

2. Tectia Server verifies the identity of the user using its own means:

   - User's public/private key pair (or certificate) for public-key authentication method.
   - Computer's public/private key pair (or certificate) for host-based authentication method.
   - Interactive challenge response via RSA Authentication Agent for SecurID submethod of keyboard interactive method.
   - Interactive authentication using RADIUS protocol for Radius submethod of keyboard interactive method.

The authentication is done in this order for technical reasons. If the second step fails, the actual authentication fails (the user is denied accessing the computer) and the user is logged off immediately. This also means that the Windows OS security log will contain a successful logon-logoff sequence even for failed authentications.

Methods that do not use native Windows account's password do not pass any credentials (password) to the Windows OS. As a consequence, Windows OS has no credentials to forward when accessing resources on Windows network.

SecurID and Radius submethods of keyboard-interactive authentication are, however, most commonly used as a two-factor authentication together with normal password authentication submethod. If a method that does not use native Windows account password is used as a second authentication method together with password authentication, the actual logon session will be created using the password provided by the user and access to network resources will work the same way as if password authentication was used alone.

Domains with Windows Server 2003 or newer domain functional level accept a new type of Kerberos request, where the service requests a ticket from the client, presenting its own credentials instead of the client's. This extension is called Service-for-User-to-Self (S4U2Self). Tectia Server is using this service to obtain the user's access token without providing a password. These Kerberos extensions allow users in domains with Windows Server 2003 or newer domain functional level to access some network resources. These resources must be configured on a domain controller to be trusted for delegation of credentials. See Section 5.5.3, Section 5.16, and Section 9.3.6 for details.

Interactive access to network resources

When tasks involving access to Windows network resources from Tectia Server's logon session are done interactively, it is usually possible to configure Tectia Server to include password authentication.

Non-interactive access to network resources

When public-key authentication (or any other method that is not using Windows native passwords) is required for non-interactive scripts, password authentication can be added as a second required authentication method and the password is passed to automated scripts via command-line option.

```
-P, --password=PASSWORD|   Set user password. Giving the PASSWORD
           file://FILE|   directly as the argument or through an
        env://VARIABLE|   environment VARIABLE is not secure. Give
        extprog://PROG    either a path to FILE containing the password
```

```
                            or a path to external program that will output
                            the password.
```

Alternatively, you can specify the path to a password file or password program in Connection Broker's configuration file. This way you can use the benefits of the authentication method of your choice and access the network resources.

5.17.1 Network Resource Access from Terminal Session

The command for working with network shares in Windows console is **net use**. When John Brown from our earlier example logs on to his company's SSH jump server using password authentication, the output of **net use** will look as follows:

```
C:\Users\johnb>net use
New connections will be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------
OK           N:        \\server1\share1          Microsoft Windows Network
The command completed successfully.
```

Now he can access the data on this disk:

```
C:\Users\johnb>N:

N:\>dir
 Volume in drive N is share1
 Volume Serial Number is 1D01-2B83

 Directory of N:\

31.03.2009  14:56    <DIR>          johns_treasure
02.04.2009  10:15                 8 johns_test.txt
12.05.2009  16:22             2 315 important_data.txt
17.06.2009  12:46            13 061 important.log
```

However, when he chooses to log on to the same server using the public-key authentication method, his experience may be as follows:

```
C:\Users\johnb>net use
New connections will be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------
Unavailable  N:        \\server1\share1          Microsoft Windows Network
The command completed successfully.
```

Now his network drive is completely unavailable:

```
C:\Users\johnb>N:
```

```
The system cannot find the drive specified.
```

Other times using the public-key authentication he will get the drive letter assigned:

```
C:\Users\johnb>net use
New connections will be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------
OK           N:        \\server1\share1          Microsoft Windows Network
The command completed successfully.


C:\Users\johnb>N:
```

His files are still inaccessible, though:

```
N:\>dir
 Volume in drive N is share1
 Volume Serial Number is 1D01-2B83

 Directory of N:\

File Not Found
```

Tectia Server attempts to restore all of the user's persistent network connections. This is done just before starting the interactive terminal session's prompt. The operation will succeed if the user provided credentials to his Windows user account. It also succeeds if all of the following is true (see Section 9.3.6):

1. If Tectia Server is running on Windows Server 2008 or newer.
2. The computer where Tectia Server is running is a member of Windows Server 2003 or newer domain functional level .
3. The user account *johnb* is a domain account in the same domain (or other domain with bi-directional trust).
4. The share being accessed is on a computer in the same domain and constrained delegation of credentials has been set up for this share on the domain controller.

However, even if the restoration of network connections failed as illustrated above, John can still try to re-connect to the drive manually since he is using the terminal session interactively. This time Windows OS will ask him to provide the user credentials of the server he is trying to access, after which John can use his files freely:

```
C:\Users\johnb>net use
New connections will be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------
```

```
Unavailable  N:        \\server1\share1          Microsoft Windows Network
The command completed successfully.


C:\Users\johnb>N:
The system cannot find the drive specified.

C:\Users\johnb>net use h: \\server1\share1
The password is invalid for \\server1\share1.

Enter the user name for 'server1': johnb
Enter the password for filer: ******************
The command completed successfully.


C:\Users\johnb>N:

N:\>dir
 Volume in drive N is share1
 Volume Serial Number is 1D01-2B83

 Directory of N:\

31.03.2009  14:56    <DIR>          johns_treasure
02.04.2009  10:15                 8 johns_test.txt
12.05.2009  16:22             2 315 important_data.txt
17.06.2009  12:46            13 061 important.log
```

5.17.2 Network Resource Access from SFTP Subsystem

Tectia Server will attempt to restore persistent network connections of the user logged on the same way also when it starts the SFTP subsystem (**sft-server-g3**). In addition to this, it will also attempt to restore connections to virtual folders that reside on network shares. However, at this point the client program communicates with the server using SFTP protocol. This protocol does not include any support for challenge response authentication because it is designed to be used over a connection that has already been authenticated. In this case the success of Windows network connection attempt depends solely on the credentials provided during the initial authentication to SSH server. Therefore, the only reliable alternative in this case is to use password authentication that uses credentials of the native Windows account.

5.17.3 Accessing Network Shares Using Another User's Account

Example: John Brown forgot to copy his test results from the folder belonging to the test account he uses for testing. It is on `\\server3\test_share` and the only user that has access to the file is *testuser*. It is possible to access the file from the user's own terminal logon session on SSH jump server:

```
C:\Users\johnb>net use \\server3\test_share <testuser's password here> /USER:testuser
The command completed successfully.


C:\Users\johnb>dir \\server3\test_share
 Volume in drive \\server3\test_share has no label.
```

```
 Volume Serial Number is 1D01-2B83

 Directory of \\server3\test_share

13.12.2011  17:15    <DIR>          .
13.12.2011  17:15    <DIR>          ..
02.04.2009  10:15             8 192 test_results.txt
```

However, it will only work if John logs on to the SSH jump server using his credentials (user name and password).

An alternative way to access the file would be to log on to SSH jump server as *testuser* using his credentials and just type:

```
C:\Users\testuser>net use \\server3\test_share

The command completed successfully.

C:\Users\testuser>dir \\server3\test_share
 Volume in drive \\server3\test_share has no label.
 Volume Serial Number is 1D01-2B83

 Directory of \\server3\test_share

13.12.2011  17:15    <DIR>          .
13.12.2011  17:15    <DIR>          ..
02.04.2009  10:15             8 192 test_results.txt
```

5.17.4 Accessing Shares on a Computer That Is Not a Member of a Domain

When the computer to be reached is not a member of the same domain as the computer with SSH server, the shares on it can still be accessed. However, the access will only work if the Windows native user account credentials are used for authentication and if the user account with the same name and password exists on the remote machine.

Accessing shares as a user with a different name works in an interactive session as described in Section 5.17.3.

5.17.5 Access to DFS Shares

Access to Distributed File System resources only works with password authentication.

5.18 Accessing Files Stored on EFS on Windows from Logon Sessions Created by Tectia Server

A prerequisite to use EFS (Encrypting File System) encrypted files from logon session created by Tectia Server is the correct configuration of EFS on the host itself. How to use the Encrypting File System, see

http://msdn.microsoft.com/en-us/library/ms995356.aspx. It should be made sure that the access works as expected for users who are logged on to the system using native Windows means (e.g., from console or via Remote Desktop).

You will be able to access EFS encrypted files from sessions created by Tectia Server only if you have logged in using credentials of the native Windows user account.

Chapter 6 System Administration

Secure system administration is the most common use case for Secure Shell. This chapter describes typical system administration settings and available auditing options of Tectia Server.

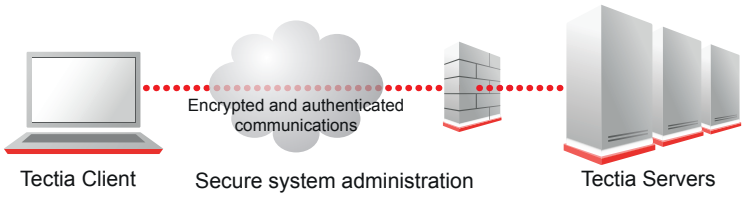


Figure 6.1. Secure system administration

6.1 Tectia Client Privileged User

The configuration of Tectia Server typically sets limitations on secure system administration. Tectia Server often resides in the DMZ. Strong two-factor authentication is often required from privileged users and connections are allowed only from certain hosts.

6.1.1 Disabling Root Login (Unix)

Restrictions on Secure Shell services, as described for non-privileged users in Section 7.1.2 and Section 8.1.2, do not prevent users with shell access to the system from setting up the equivalent services.

It is also possible to limit users with administrative privileges to predefined commands if shell access is not needed.

Shell access is often desired for remote administration of the server computer. It is recommended to have users log in first to their non-privileged user accounts and once logged in elevate their rights using `sudo` or `su` especially if the `root` account is used instead of individual administrator accounts.

The following configuration setup prevents logging directly in to the privileged accounts:

```
<authentication-methods>
```

```
  <authentication action="deny">
    <selector>
      <user-privileged value="yes" />
    </selector>
  </authentication>
  ...
</authentication-methods>
```

6.1.2 Restricting Connections

Tectia Server can be configured to reject connection attempts from unknown hosts. For example the following allows connections only from the internal network `10.1.0.0/8` IP addresses and from an external host with the IP address `195.20.116.1`:

```
<connections>
  <connection action="allow">
    <selector>
      <ip address="10.1.0.0/8" />
      <ip address="195.20.116.1" />
    </selector>
  </connection>
  <connection action="deny" />
</connections>
```

Using the **Tectia Server Configuration** GUI, the same settings can be made under the **Connections and Encryption** page, on the **Selectors** tab. See the section called “Editing Connection Rules”.

For information on the selectors, see Section 4.2.2.

On systems with several network interfaces, Tectia Server can also be bound to a specific network interface so that the server can be only accessed from the intended network. For example, the following will bind the listener to address `10.1.60.25` using the Secure Shell default port 22:

```
<params>
  <listener id="intranet" address="10.1.60.25" />
  ...
</params>
```

Using the **Tectia Server Configuration** GUI, this can be set on the **Network** page. See Section 4.1.7.

6.1.3 Chrooting (Unix)

By default, file access by users is restricted by the file system access controls. On Unix, access can be further restricted with the usage of the `chroot` attribute. The `chroot` attribute can be used with the `subsystem`, `terminal`, and `command` elements.

The `chroot` attribute must be a directory path. Values `%username%`, `%homedir%`, and `%hostname%` will be substituted with the user name, user's home directory, and the FQDN of the connected client, respectively. The values are read from the environment variables defined in the system.

The following sections give instructions on chrooting the terminal, remote commands, and subsystems.

Chrooting Terminal

An example of chrooting the terminal is shown below:

```
<services>
  <rule>
    <terminal action="allow" chroot="%homedir%" />
   ...
  </rule>
  ...
</services>
```

When users are restricted to the chrooted environment, they cannot access the normal shell binary. This means that the shell specified in the `/etc/passwd` file for the user has to be present in the equivalent place under the chrooted directory. For example, if `/etc/passwd` lists `/bin/bash` as the shell and the user is chrooted to the home directory, a statically linked `%homedir%/bin/bash` should exist.

If the user's shell is dynamically linked, you must make sure that the required shared libraries are also in the chrooted environment. You can resolve the dependencies with the `ldd` command:

```
$ ldd /bin/bash
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40026000)
        libdl.so.2 => /lib/libdl.so.2 (0x4002a000)
        libc.so.6 => /lib/libc.so.6 (0x4002d000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
$ ls lib
ld-linux.so.2  libc.so.6  libdl.so.2  libtermcap.so.2  libtermcap.so.2.0.8
```

Also note that shared libraries can have other dependencies:

```
$ ldd libtermcap.so.2.0.8
        libc.so.6 => /lib/libc.so.6 (0x40017000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
```

The user's environment might also need some other tools, such as:

- `ls` for listing files
- `stty` for setting tty modes

You might also need some device files under the user's virtual root directory. At least a `/dev/null` file is needed on Linux. You can create it as follows:

```
$ mkdir dev
$ cd dev
$ ls -l /dev/null
crw-rw-rw-    1 root     root       1,   3 Jan 30  2003 /dev/null
$ mknod null c 1 3
$ chmod go+w null
$ ls -l null
```

Chrooting Remote Commands

An example of chrooting a remote command is shown below:

```
<services>
  <rule>
    <command application="date" action="allow" chroot="%homedir%" />
   ...
  </rule>
  ...
</services>
```

Now, the user is restricted to the home directory when running `sshg3` with the remote command `date`:

```
$ sshg3 user@server date
```

The command to be run has to be statically linked and available under the chrooted environment. In the above example when the user is chrooted to the home directory, a statically linked `date` command should exist in `%homedir%/bin`.

If the command is dynamically linked, you must make sure that the required shared libraries are also in the chrooted environment. See the section called “Chrooting Terminal” above.

Chrooting SFTP

By default, file access by the user using the SFTP subsystem is restricted by the file system access controls. On Unix, access can be further restricted with the usage of the `chroot` attribute.

An example of `chroot` usage is shown below:

```
<services>
  <rule>
    <subsystem type="sftp"
               application="sft-server-g3"
               action="allow"
               chroot="/home/%username%" />
   ...
  </rule>
  ...
</services>
```

Here `%username%` will be substituted with the current user name. For a user named `example`, the path would be `/home/example`. During an SFTP session, the user is now restricted to this directory (and its subdirectories).

Note

Chrooting the SFTP subsystem affects both SFTP and SCP2 operations to the server, but it does NOT affect legacy OpenSSH-style SCP operations. To chroot also OpenSSH SCP (version 8 and older), you should chroot the `scp1-compat-srv` command. For instructions on chrooting commands, see the section called “Chrooting Remote Commands”. If you do not need to allow OpenSSH SCP, you can disable all remote commands as described in the section called “Disabling Remote Commands”.

6.1.4 Forced Commands

A forced command causes a specified application to run automatically when the user logs in. All other applications are implicitly denied.

If you have maintenance jobs requiring non-interactive access to your server, use public-key authentication and forced commands. This way, if the private key is compromised, the public key cannot be used to perform anything other than the predetermined command on the server. This is, of course, also bad, but it would be worse if the malicious attacker would have unrestricted access to the machine.

Do not use the root (administrator) account for jobs where it is not absolutely necessary.

You can set up a forced command in the `ssh-server-config.xml` file.

```
<services>
  <rule group="backup">
    <terminal action="deny" />
    <!-- This account is only used to backup the disk drive. -->
    <subsystem type="sftp" application="sft-server-g3" action="deny" />
    <command application="dd if=/dev/hda" action="forced" />
    <tunnel-local action="deny" />
    <tunnel-remote action="deny" />
  </rule>
  ...
</services>
```

This would, on a successful login as the group `backup`, force a backup job to start.

Using the **Tectia Server Configuration** GUI, the same setting can be made under the **Services** page on the **Commands** tab. See the section called “Commands”.

You can also use the command that was given on the `sshg3` command line:

```
<services>
  <rule group="admin">
    <command application="echo $SSH2_ORIGINAL_COMMAND" action="forced" />
    ...
  </rule>
  ...
</services>
```

Running `sshg3`:

```
$ sshg3 localhost kukkuu
kukkuu
$
```

6.2 Auditing

Tectia Server logs events in the `syslog` on Unix and in the Windows Event Log on Windows. Logging (auditing) is very important for security. You should check your logs often, or use tools to analyze them. From the logs, you can see, for example, whether unauthorized access has been attempted, and take further action if needed. For example, you could set the hosts from which the attempts have been made as denied,

or drop the packets from the domain completely at your firewall. The logs also provide troubleshooting information.

The log events are classified in seven levels, in decreasing order of importance:

Security failure (Windows only)

A user tried to log on but failed.

Security success (Windows only)

A user logged successfully on.

Critical (Unix only)

A critical problem has occurred. By default, this is not used by Tectia Server.

Error

A serious problem has occurred, preventing the intended operation from completing successfully.

Warning

A problem has occurred, but the operation can continue.

Notice (Unix only)

An action has been done.

Informational

Extra troubleshooting information.

6.2.1 Notification

It is recommended to notify the users before they decide to log in that their actions are logged. In some jurisdictions this is required.

To display, for example, the following text to the users before login, you can define a `banner-message` element in the `ssh-server-config.xml` file or with the **Tectia Server Configuration** tool. See the section called “The `authentication-methods` Block” or Section 4.1.2 for more information.

```
Unauthorized use of this system is prohibited.
All actions are logged.
```

6.2.2 Customizing Logging

Tectia Server allows customizing the severity and facility of different logging events. The events have reasonable default values, which are used if no explicit logging settings are made.

The logging settings are made in the `logging` element of the `ssh-server-config.xml` file or with the **Tectia Server Configuration** tool. See the section called “The `params` Block” or Section 4.1.8 for more information.

The default logging settings of Tectia Server in the `ssh-server-config-default.xml` file are shown below:

```
<logging>
  <log-events facility="auth" severity="informational">
    Auth_method_success Auth_method_failure Auth_methods_completed
    Auth_methods_available Hostbased_auth_warning
    Publickey_auth_warning Publickey_auth_success GSSAPI_auth_warning
    Keyboard_interactive_pam_auth_warning
    Keyboard_interactive_radius_auth_warning
    Keyboard_interactive_securid_auth_warning
    GSSAPI_auth_success
    Keyboard_interactive_pam_auth_success
    Keyboard_interactive_radius_auth_success
    Keyboard_interactive_password_auth_success
    Keyboard_interactive_securid_auth_success
  </log-events>
  <log-events facility="auth" severity="warning">
    Hostbased_auth_error Publickey_auth_error GSSAPI_auth_error
    Keyboard_interactive_pam_auth_error
    Keyboard_interactive_radius_auth_error
    Keyboard_interactive_password_auth_error
    Keyboard_interactive_securid_auth_error
  </log-events>
  <log-events facility="daemon" severity="error">
    Server_start_failed
  </log-events>
  <log-events facility="daemon" severity="notice">
    Server_listener_failed Server_listener_started
    Server_listener_stopped Server_reconfig_finished
    Server_reconfig_started Server_stopping Server_running
    Server_starting
  </log-events>
  <log-events facility="daemon" severity="warning">
    Servant_exited Servant_error
  </log-events>
  <log-events facility="normal" severity="informational">
    Algorithm_negotiation_success Certificate_validation_success
    Certificate_validation_failure Key_store_create
    Key_store_destroy Key_store_add_provider Key_store_decrypt
    Key_store_sign Key_store_sign_digest Logout Disconnect
    Channel_open_failure Session_channel_open
    Session_channel_close Forwarding_channel_open
    Forwarding_channel_open Forwarding_channel_close
    Forwarding_listener_open Forwarding_listener_close
    Auth_listener_open Auth_listener_close Auth_channel_open
    Auth_channel_close
  </log-events>
  <log-events facility="normal" severity="security-failure">
    Connection_denied Login_failure
  </log-events>
  <log-events facility="normal" severity="security-success">
    Connect Login_success
  </log-events>
```

```
  <log-events facility="normal" severity="warning">
    Algorithm_negotiation_failure KEX_failure
    Key_store_create_failed Key_store_add_provider_failed
    Key_store_decrypt_failed Key_store_sign_failed
    Key_store_sign_digest_failed
  </log-events>
</logging>
```

For a description of the log events, see Appendix D.

6.2.3 Auditing with Solaris BSM

On Solaris platforms, Basic Security Module (BSM) can be used to audit Secure Shell log-in (both failed and successful) and log-out events.

The log-in events are audited with the event ID 34543 (`AUE_tectia`) and the log-outs with event ID `AUE_logout`.

When auditing `AUE_tectia` events, add the following line to `/etc/security/audit_event`:

```
34543:AUE_tectia:login - ssh:lo
```

To prevent clashes with other BSM-aware third-party applications, you can change the `AUE_tectia` event ID to a unique one by exporting the environment variable `SSH_BSM_AUDIT_EVENT_ID=<event_id>` before you start Tectia Server.

Chapter 7 File Transfer

This chapter gives the typical Tectia Server settings when it is used for secure file transfer.

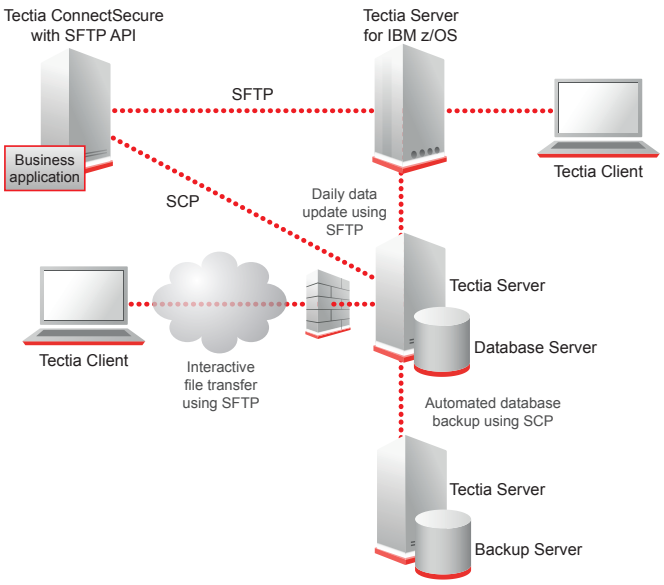


Figure 7.1. Secure file transfer

Tectia Server supports the basic secure file transfer functionalities provided by Tectia Client, but also the more advanced secure file transfer functions provided by Tectia ConnectSecure. These include the enhanced file transfer (EFT) functionalities, such as checkpoint/restart for the transfer of very large files, streaming for high-speed file transfers, and C and Java APIs for customization.

For more information on the enhanced file transfer features available with Tectia ConnectSecure, see *Tectia Client/Server Product Description*, *Tectia ConnectSecure Administrator Manual*, and the documentation for the C and Java APIs (included in the installation package).

7.1 Tectia Client File Transfer User

When Tectia Server is used for automated file transfer, separate user accounts can be created for the file transfer users. Non-interactive authentication with public keys and scripted commands can be set for these accounts.

7.1.1 Encryption and Authentication Methods

To increase file transfer speed, the CryptiCore algorithm should be enabled on the server (if available). To allow non-interactive authentication, public keys can be used.

Enabling CryptiCore

The CryptiCore algorithm is supported on x86-based processor architectures. It allows increased file transfer speeds for large file transfers.

To use CryptiCore, include the following in the `ssh-server-config.xml` file:

```
<connections>
  <connection action="allow" tcp-keepalive="no">
    <rekey seconds="3600" bytes="1000000000" />
    <cipher name="crypticore128@ssh.com" />
    <mac name="crypticore-mac@ssh.com" />
  </connection>
</connections>
```

Using the **Tectia Server Configuration** GUI, this can be set under the **Connections and Encryption** page, on the **Parameters** tab. See the section called “Parameters”.

Enabling Public-Key Authentication

To enable public-key authentication on the server, include the following in the `ssh-server-config.xml` file:

```
<authentication-methods login-grace-time="600">
  <banner-message />
  <auth-file-modes strict="yes" mask-bits="022" />
  <authentication>
    <auth-publickey />
  </authentication>
</authentication-methods>
```

The `auth-file-modes` element should be set to strict. This specifies that Tectia Server on Unix checks the permissions and ownership of the user's key files used for public-key authentication.

Using the **Tectia Server Configuration** GUI, the same settings can be made under the **Authentication** page, on the **Parameters** tab. See the section called “Parameters”.

Note however, that the `auth-file-modes` option is not available on Windows, because strict host key checking is always used on Windows.

7.1.2 Restricting Services

If Tectia Server is used for file transfer only, it is advisable to disable remote commands, tunneling, and terminal access to the server.

On Unix, it is also possible to chroot SFTP. For instructions on that, see the section called “Chrooting SFTP”.

Enabling the SFTP Subsystem

The secure file transfer subsystem can be defined in the `ssh-server-config.xml` file:

```
<services>
  <rule>
    <subsystem type="sftp" application="sft-server-g3" />
   ...
  </rule>
  ...
</services>
```

Using the **Tectia Server Configuration** GUI, this can be set under the **Services** page, on the **SFTP** tab. See the section called “SFTP”.

Disabling Tunneling

If you are sure you or your users do not need to create tunnels (possibly going around firewall restrictions or such), you can disable tunneling (port forwarding) altogether by adding the following to the `ssh-server-config.xml` file:

```
<services>
  <rule>
   <tunnel-local action="deny" />
   <tunnel-remote action="deny" />
   ...
  </rule>
  ...
</services>
```

Using the **Tectia Server Configuration** GUI, these can be set under the **Services** page, on the **Basic** tab. See the section called “Basic”.

If you need more fine-grained control, you can define user groups in the `services` block and apply the restrictions only to the specified groups.

Tunneling restrictions can be further defined with the `src`, `dst`, and `listen` elements. See Chapter 8 for more information.

Disabling Terminal Access

If you only want to enable file transfers or tunneling for users in group `remote-access`, you can disable terminal access by adding the following to the `ssh-server-config.xml` file:

```
<services>
  <rule group="remote-access">
    <terminal action="deny" />
```

```
    ...
  </rule>
  ...
</services>
```

Using the **Tectia Server Configuration** GUI, this can be set under the **Services** page, on the **Basic** tab. See the section called “Basic”.

This setting denies also X11 and agent forwarding and shell commands for the specified group (unless some commands are explicitly allowed).

The users will be able to use SFTP and other subsystems defined in the Tectia Server configuration. Any other "exec" and "shell" requests will be denied for the users. This includes forced commands with public keys described in Section 6.1.4 and the legacy style password changing when performed as forced command.

Disabling Remote Commands

If you wish to restrict what users can do with remote commands, we recommend that you set `deny` as the default action, and then allow only some specific remote commands, if any. This way you do not accidentally leave some unnoticed commands as allowed.

When terminal access is denied, also the remote commands are denied unless you explicitly define the commands as allowed or as forced. With the terminal access denied, it is advisable to allow only some specific commands. If all commands are allowed, the remote command users can perform most of the things they could do with the terminal access allowed. For more information, see **command** .

Note that restrictions on remote commands apply also to OpenSSH-style SCP operations to the server.

You can disable remote commands totally by adding the following settings to the `ssh-server-config.xml` file:

```
<services>
  <rule>
   <command action="deny" />
   <terminal action="deny" />
   ...
  </rule>
  ...
</services>
```

If you need more fine-grained control, you can define user groups in the `services` block and apply the restrictions only to the specified groups.

Using the **Tectia Server Configuration** GUI, remote commands can be disabled under the **Services** page, on the **Basic** tab. See the section called “Basic”.

Defining SFTP Virtual Folders (Windows)

Virtual folders can be used to restrict the folders the user is able to access via SFTP and SCP2. The operating system file permissions are also enforced. The SFTP user must have intended read and write permissions, and at least read and list directory permissions for the parent folder of the virtual folder.

By default, if no virtual folders are explicitly defined in the configuration file, the user can access all drives via SFTP and SCP2 operations, the user's SFTP session starts in the `%USERPROFILE%` directory, and that is the target directory for SCP2 operations.

When any virtual folders are defined, the user access is limited to the specified folders only. Note that the user's home directory must be under one of the defined virtual folders.

It is also possible to change the SFTP starting directory and the target directory of the SCP2 operations by defining the `home` attribute in the configuration file. Its value can also contain special strings which are expanded by Tectia Server. These strings are `%username%` (user's login name), `%username-without-domain%` (user's login name without the domain part), `%homedir%` (user's home directory), and `%hostname%` (the name of the host the user is logging from, reverse mapped from the IP).

Note

Some special characters such as a slash "/" and a backslash "\" cannot be used in the name of the virtual folder (for example `network_share`) displayed to the SFTP user.

In case a trailing dollar sign $ is used in the virtual folder destination path (for example `\\server\share$`), the sign has to be escaped as follows:

```
network_share=\\server\share$$
```

The following example sets the starting directory to a user-specific subdirectory under `C:\SFTP`.

```
<services>
  <rule>
    <subsystem type="sftp" application="sft-server-g3" action="allow">
      <attribute name="virtual-folder"
                 value="ftp_home=C:\SFTP\%username-without-domain%" />
      <attribute name="home" value="C:\SFTP\%username-without-domain%" />
    </subsystem>
    ...
  </rule>
  ...
</services>
```

If the `home` attribute is included in the configuration file but it is given an empty value, or if a directory that is denied by the virtual folder settings is specified, the session will start in the virtual SFTP root directory.

```
<services>
  <rule>
    <subsystem type="sftp" application="sft-server-g3" action="allow">
      <attribute name="virtual-folder" value="C:=C:\" />
      <attribute name="home" value="" />
    </subsystem>
   ...
  </rule>
  ...
</services>
```

Note

The virtual SFTP root folder is not an actual directory on disk and no files can be written there.

To define custom virtual folders, the `virtual-folder` attribute can be used in the configuration file. If any virtual folders are defined, the default drive letters are not used. If you still want to use the drive letters, they need to be defined in the configuration file.

The value of virtual folder can contain the same special strings as the value of home (`%username%`, `%username-without-domain%`, `%homedir%`, and `%hostname%`).

The following example allows access to the `C:` drive and a user-specific subdirectory under the `SFTP` parent folder on the `D:` drive (when a user changes directory to `D:`, he is actually directed to the user-specific directory). The session starts in the virtual SFTP root folder. No other directory can be accessed via SFTP.

```
<services>
  <rule>
    <subsystem type="sftp" application="sft-server-g3" action="allow">
      <attribute name="virtual-folder" value="C:=C:\" />
      <attribute name="virtual-folder" value="D:=D:\SFTP\%username%" />
      <attribute name="home" value="" />
    </subsystem>
   ...
  </rule>
  ...
</services>
```

In the **Tectia Server Configuration** tool, virtual folders can be set under the **Services** page on the **SFTP** tab. See the section called “SFTP”. The settings in the example above are shown in Figure 7.2.

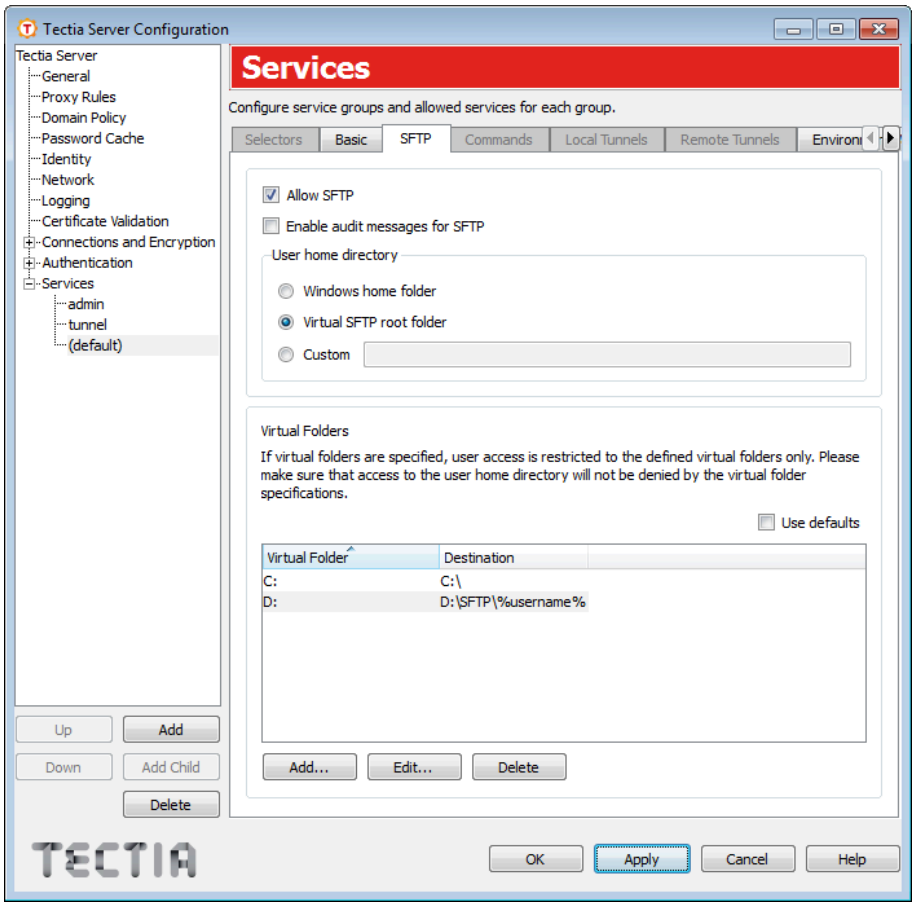


Figure 7.2. Defining virtual folders

7.1.3 Settings on the Client Side

For example, the following configuration can be used in the `ssh-broker-config.xml` file:

```
<profile name="sftexa"
             id="id1"
             host="sftexa.ssh.com"
             port="12345"
             connect-on-startup="no"
             user="sftuser">
      <ciphers>
        <cipher name="crypticore128@ssh.com" />
      </ciphers>
      <macs>
        <mac name="crypticore-mac@ssh.com" />
      </macs>
      <authentication-methods>
        <auth-publickey>
      </authentication-methods>
      <compression name="none"/>
      <server-banners visible="no" />
</profile>
```

To enable non-interactive authentication, the private key on the Client is stored with a NULL passphrase. It is important that the key directory and the key file have the correct permissions (for example, 700). For more information, see Section 5.6.

7.2 Automated File Transfer Script

This section gives an example of setting up automated file transfer between Tectia Client and Server hosts using scripts.

The following example script first transfers a file from Tectia Client to Tectia Server and then transfers the file back. The script logs the command and the return values to a file.

```
#!/bin/bash

DATE=`date +%d.%m.%Y-%H.%M`
SRV=sftexa

#scpg3 put
echo "/opt/tectia/bin/scpg3 -B -q testfile $SRV:test" >> scpg3_put_$DATE
/opt/tectia/bin/scpg3 -B -q testfile.dat $SRV:test
echo $? >> scpg3_put_$DATE

#scpg3 get
echo "/opt/tectia/bin/scpg3 -B -q $SRV:test test" >> scpg3_get_$DATE
/opt/tectia/bin/scpg3 -B -q $SRV:test test
echo $? >> scpg3_get_$DATE
```

The script can be set to run as a forced command. See Section 6.1.4.

Chapter 8 Tunneling

Tunneling is a way to forward otherwise unsecured TCP traffic through Secure Shell in encrypted format. 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, 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 (incoming) tunnels, the ports under 1024 (the well-known service ports) are not allowed for the regular users, but are available only for system administrators (root privileges).

There are two basic kinds of tunnels: local (outgoing) and remote (incoming). X11 forwarding and agent forwarding are special cases of a remote tunnel.

Tectia Client and Tectia ConnectSecure both provide basic tunneling functionalities. Tectia ConnectSecure also provides transparent TCP tunneling and other more advanced tunneling features, see *Tectia ConnectSecure Administrator Manual*.

This chapter gives an example of the Tectia Server settings for a transparent TCP tunneling case and describes the different tunneling options available together with Tectia Client and Tectia ConnectSecure.

8.1 Transparent TCP Tunneling from Server Perspective

Tectia ConnectSecure provides transparent TCP tunneling of applications. It can connect to any Secure Shell server complient with IETF SSH version 2. Tectia Server and Tectia Server for IBM z/OS both support transparent TCP tunneling. In this document, we handle the settings of Tectia Server.

The ConnectSecure users must be able to log in to an existing user account, preferably a non-privileged user account, on the server.

Users can have their own user accounts. If the Windows login name can be used also as the server-side login name, the variable `%USERNAME%` can be conveniently used in the configuration of Tectia ConnectSecure.

Most of the user authentication methods supported by Tectia Server can be used with transparent TCP Tunneling. The authentication methods include password, any keyboard-interactive methods such as SecurID or RADIUS, public-key authentication with certificates on smart cards, and GSSAPI if ConnectSecure and the server computers are part of the same Windows domain, or Tectia Server can perform initial login to MIT Kerberos realm on behalf of the ConnectSecure user.

User interaction is required for the keyboard-interactive authentication methods and typically at least the first time when the private key stored on a smart card is accessed in public-key authentication. For details on the user authentication methods, see Chapter 5.

8.1.1 Using a Shared Account

In case the tunneled applications provide sufficient user authentication, it is possible to use a shared user account, for example with a shared password, not requiring user interaction. Note that the shared account and password must only be used for tunneling, as the account is common to several users and the shared password is stored as plaintext in the Connection Broker configuration file.

See the operating system documentation for instructions on how to create a new user account, for example `tunnel`, with minimal privileges. It is very important that the shared user account is properly configured on the operating-system level.

The user should be denied at least shell access and the file system permissions should be restricted. This is done as a precaution in case the user is able to access the system using some other means than Secure Shell.

To deny shell access on the operating-system level, you can set the user's shell to `/bin/false` or use a script that can also inform the user of the situation:

For example, you could have the following saved to name `/bin/no-shell`:

```
#!/bin/sh

echo "Shell access to this account has been disabled."
exit 1
```

8.1.2 Restricting Services

In this example, the user `tunnel` is restricted to tunneling services while other users have terminal access. All users are denied file transfer service and X11 and agent forwarding.

Note that the users with terminal (shell) access are restricted only in the Tectia Server configuration and can, for example, set up their own port forwardings. For more information, see Section 6.1.

Tunneling

Transparent TCP tunneling uses only local tunnels. The tunnels are established based on the configuration of the application being tunneled. For details on the tunneling principles, see Section 8.2.

The following configuration options of Tectia Server will deny remote tunnels (remote port forwarding) and allow local tunnels (local port forwarding) for all users for example to `http://webserver.example.com` or `https://webserver.example.com`.

```
<services>
  <rule>
    <tunnel-local action="allow">
      <dst fqdn="*.example.com" port="80" />
      <dst fqdn="*.example.com" port="443" />
    </tunnel-local>
    <tunnel-local action="deny" />
    <tunnel-remote action="deny" />
    ...
  </rule>
</services>
```

Disabling Terminal Access

The following configuration options of Tectia Server will deny terminal access from users in group `tunnel`.

```
<services>
  <group name="tunnel">
    <selector>
      <user name="tunnel" />
    </selector>
  </group>
  <rule group="tunnel">
    <terminal action="deny" />
    <subsystem type="sftp" application="sft-server-g3" action="deny" />
    <command action="forced" application="no-shell" />
    ...
  </rule>
  ...
</services>
```

Denying terminal denies also X11 and agent forwarding and shell commands (unless some commands are explicitly allowed).

The `command` action in this example provides an alternative method of informing the user of denied shell access using the `/bin/no-shell` script introduced in Section 8.1.1.

This method can be used if the risk of gaining access via other means than Secure Shell can be eliminated. This way, each user's shell does not have to be set separately, and the setting can be easily scaled to several users.

Using the **Tectia Server Configuration** GUI, the similar settings can be made under the **Services** page on the **Basic** tab. See the section called “Basic”.

Disabling File Transfers

To deny all users the access to the SFTP server, change the default SFTP subsystem configuration option of Tectia Server to:

```
...
  <rule>
    ...
    <subsystem type="sftp" action="deny" />
     ...
  </rule>
  ...
```

Using the **Tectia Server Configuration** GUI, this can be set under the **Services** page on the **SFTP** tab. See the section called “SFTP”.

8.2 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
```

Setting up local tunneling allocates a listener 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. The connection from the server onwards will not be secure, it is a normal TCP connection.

For example, when you use Tectia Client on the command line, and issue the following command, all traffic coming to port 1234 on the client will be forwarded to port 23 on the server. See Figure 8.1.

```
sshclient$ sshg3 -L 1234:localhost:23 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`).

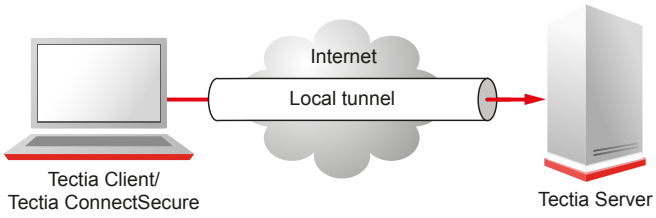


Figure 8.1. Simple local tunnel

To use the tunnel, the application to be tunneled is set to connect to the local listener port instead of connecting to the server directly. Tectia Client or ConnectSecure forwards the connection securely to the remote server.

If you have three hosts, for example, `sshclient`, `sshserver`, and `imapserver`, and you forward the traffic coming to the `sshclient` port 143 to the `imapserver` port 143, only the connection between `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 8.2 shows an example where the Secure Shell server resides in the DMZ network. The connection is encrypted from the Secure Shell client to the Secure Shell server and continues unencrypted in the corporate network to the IMAP server.

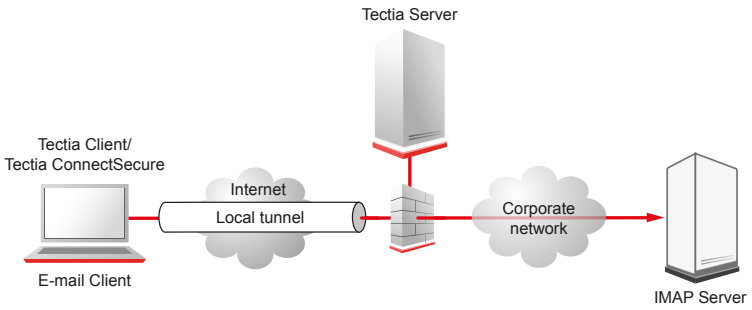


Figure 8.2. Local tunnel to an IMAP server

With transparent TCP tunneling active, there is no need to separately configure application software to use local ports to set up the tunnels. The applications to be tunneled are defined in the Connection Broker configuration (**Filter Rules**). The transparent TCP tunneling feature automatically captures the defined applications and the Connection Broker creates Secure Shell tunnels to the defined Tectia Server.

By default, local tunnels are allowed to all addresses for all users. The default setting equals the following in the `ssh-server-config.xml` file:

```
<services>
  <rule>
    <tunnel-local action="allow" />
    ...
  </rule>
</services>
```

The connections can be restricted by specifying allowed addresses with the `src` and `dst` elements. If any addresses are specified as allowed, local tunnels to all other addresses are implicitly denied. See Section 8.2.1 for usage examples.

Using the **Tectia Server Configuration** GUI, the tunneling settings are made under the **Services** page on the **Local Tunnels** tab. See the section called “Local Tunnels”.

8.2.1 Local Tunneling Rule Examples

This section gives examples on using the local tunneling rules in the `ssh-server-config.xml` file.

Figure 8.3 shows the different hosts and ports involved in local port forwarding.

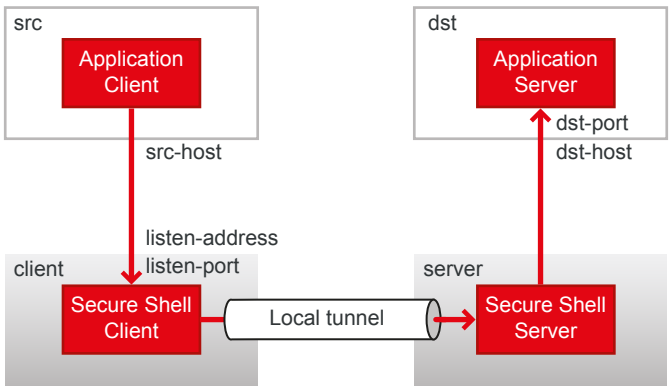


Figure 8.3. Local tunneling terminology

Allow Rules

The following configuration allows tunnels to `imap.example.com` ports 143 and 993. If this is the only `tunnel-local` rule, tunnels to all other addresses are denied:

```
<rule>
  <tunnel-local action="allow">
    <dst fqdn="imap.example.com" port="143" />
    <dst fqdn="imap.example.com" port="993" />
  </tunnel-local>
...
</rule>
```

Opening the tunnel with a matching IP address should also work. Opening any other tunnel with IP address is denied.

The following configuration allows tunnels to 192.0.2.99 port 143. If this is the only `tunnel-local` rule, tunnels to all other addresses are denied:

```
<rule>
  <tunnel-local action="allow">
    <dst address="192.0.2.99" port="143" />
  </tunnel-local>
...
</rule>
```

Opening the tunnel with a matching FQDN should also work. Opening any other tunnel with FQDN is denied.

The following configuration allows tunnels only to the server host itself (any port in the loopback interface):

```
<rule>
  <tunnel-local action="allow">
    <dst fqdn="localhost" />
  </tunnel-local>
...
```

```
</rule>
```

Note that the client must request tunneling to the internal loopback interface, not the external interface of the server host. For example, if transparent TCP tunneling is used on the client side and the external interface of the server host has IP address `192.168.92.250`, then the following configuration allows tunnels only to the server host itself (any port in the loopback interface or the external interface):

```
<rule>
  <tunnel-local action="allow">
    <dst fqdn="localhost" />
    <dst address="192.168.92.250" />
  </tunnel-local>
...
</rule>
```

By default, Tectia Client binds the listener to the internal loopback interface on the client side allowing only local connections. If the `allow-relay` option is used on the Tectia Client, all client-side interfaces are used, allowing connections from other hosts unless Tectia Server denies this with the following configuration:

```
<rule>
  <tunnel-local action="allow">
    <src address="127.0.0.1" />
  </tunnel-local>
...
</rule>
```

The configuration allows tunnels originating from the client host only. However, restrictions based on the source address of local port forwarding are normally not reliable because the client can forge the source address. A configuration like this can be used only if the client can be trusted (for example, if it is administered by yourself).

The following configuration allows tunnels originating from the `example.com` domain to the `appserver.example.com` ports 2000-9000. The `"*"` wildcard character matches all host names. As in the previous example, this configuration should be used only if the client can be trusted:

```
<rule>
  <tunnel-local action="allow">
    <src fqdn="*.example.com" />
    <dst fqdn="appserver.example.com" port="2000-9000" />
  </tunnel-local>
...
</rule>
```

The following configuration defines that Tectia Server uses the Python script `tunneling_control_script.py` as an external application to verify local tunneling connections:

```
<rule>
  <tunnel-local action="allow">
    <mapper command="python tunneling_control_script.py" timeout="25"/>
  </tunnel-local>
...
</rule>
```

Tectia Server uses the Tectia Mapper Protocol (for more information, see Appendix E) to communicate with the external application. In this example the external application is used to check if user `jbrown` from IP address `203.0.113.1` port `36388` is allowed to tunnel to IP address `198.51.100.1` on port `12346`. Tectia Server sends the following messages to the application:

```
version:1
request:1
user=45678:jbrown
user-privileged=false
{tunnel-src}addr-ip=203.0.113.1
{tunnel-src}port=36388
{tunnel-src}addr-fqdn=abc.engineering.example.com
{tunnel-dst}addr-ip=198.51.100.1
{tunnel-dst}port=12346
{tunnel-dst}addr-fqdn=def.marketing.example.com
end-of-request:1
```

The external application sends back a positive response:

```
version:1
request:1
success:
```

Deny Rules

The following configuration denies tunnels to the `192.168.23.0/24` domain. If this is the only `tunnel-local` rule, tunnels to all other addresses are allowed. If tunneling is attempted using a FQDN, the server will attempt to match to the IP addresses using a DNS lookup:

```
<rule>
  <tunnel-local action="deny">
    <dst address="192.168.23.0/24" />
  </tunnel-local>
...
</rule>
```

The following configuration denies tunnels to the `*.forbidden.example` domain. If this is the only `tunnel-local` rule, tunnels to all other addresses are allowed. If tunneling is attempted using an IP address, the server will attempt to match to the FQDN using a reverse DNS lookup. However, if this lookup fails, tunneling using IP address is allowed:

```
<rule>
  <tunnel-local action="deny">
    <dst fqdn="*.forbidden.example" />
  </tunnel-local>
...
</rule>
```

To explicitly deny this, use a configuration similar to the following one:

```
<rule>
  <tunnel-local action="deny">
    <dst fqdn="*.forbidden.example" />
```

```
    <dst address="0.0.0.0/32" />
  </tunnel-local>
...
</rule>
```

This denies all local tunneling attempts using IP addresses.

Note that the tunneling rules are not intended to replace other access control mechanisms. If strict access control is required, it should be implemented at the application servers.

8.3 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 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.

For example, if you issue the following command, all traffic which comes to port `1234` on the server will be forwarded to port `23` on the client. See Figure 8.4.

```
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.

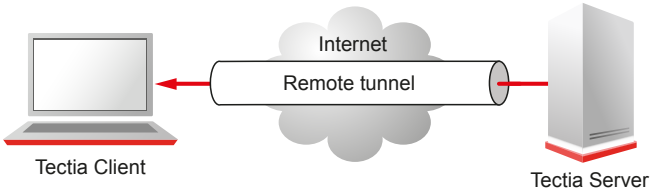


Figure 8.4. Remote tunnel

By default, remote tunnels are allowed from all addresses for all users. The default setting equals the following in the `ssh-server-config.xml` file:

```
<services>
  <rule>
    <tunnel-remote action="allow" />
    ...
  </rule>
```

```
</services>
```

The connections can be restricted by specifying allowed addresses with the `src` and `listen` elements. If any addresses are specified as allowed, remote tunnels to all other addresses are implicitly denied. See Section 8.3.1 for usage examples.

The server starts listeners according to the current address family settings. For example, if the server is configured for IPv4 only, the following command will start listener on port `2001`, address `127.0.0.1`:

```
client$ sshg3 -R 2001:localhost:2002 user@server
```

If the address family is `any`, two listeners will be started, on address `::1` and `127.0.0.1` on the same port `2001`.

Using the **Tectia Server Configuration** GUI, the tunneling settings are made under the **Services** page on the **Remote Tunnels** tab. See the section called “Remote Tunnels”.

8.3.1 Remote Tunneling Rule Examples

This section gives examples on using the remote tunneling rules in the `ssh-server-config.xml` file.

Figure 8.5 shows the different hosts and ports involved in remote port forwarding.

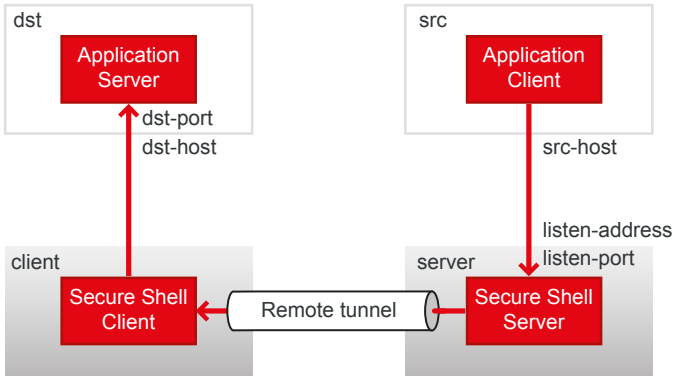


Figure 8.5. Remote tunneling terminology

Allow Rules

The following configuration allows opening a listener to port 8765 on the interface `10.1.60.16` on the server and allows connections to it from all addresses. If this is the only `tunnel-remote` rule, attempts to open remote port forwarding to other interfaces or other ports will be denied:

```
<rule>
  <tunnel-remote action="allow">
    <listen address="10.1.60.16" port="8765" />
  </tunnel-remote>
...
</rule>
```

The following configuration allows opening any port on any interface on the server but allows connections only from the listed addresses:

```
<rule>
  <tunnel-remote action="allow">
    <src fqdn="alpha.example.com" />
    <src fqdn="beta.example.com" />
  </tunnel-remote>
...
</rule>
```

By default, only users with administrative privileges can create listeners to privileged ports (below 1024). To allow any user to create listeners to privileged ports, enable the `disable-privilege-check` attribute, similar to the following:

```
<rule>
  <tunnel-remote disable-privilege-check="yes" action="allow">
  ...
  </tunnel-remote>
</rule>
```

Deny Rules

The following configuration denies opening ports 1-9000 on the server. If this is the only `tunnel-remote` rule, it allows opening all other ports:

```
<rule>
  <tunnel-remote action="deny">
    <listen port="1-9000" />
  </tunnel-remote>
...
</rule>
```

The following configuration denies connections to ports 1-9000 from the listed addresses. However, listeners can be opened to these ports (with ports 1-1023 restricted to admin users only) and all other addresses can connect to them. If this is the only `tunnel-remote` rule, it allows opening all other ports and allows connections to them from all other addresses:

```
<rule>
  <tunnel-remote action="deny">
    <listen port="1-9000" />
    <src fqdn="gamma.example.com" />
    <src fqdn="delta.example.com" />
  </tunnel-remote>
...
</rule>
```

A rule like the above probably does not have any practical use. Nevertheless, it is shown here as an example of the rule logic.

8.4 X11 Forwarding (Unix)

X11 forwarding is a special case of remote tunneling.

Tectia Server supports X11 forwarding on Unix platforms. Tectia Client and ConnectSecure support X11 forwarding on both Unix and Windows platforms.

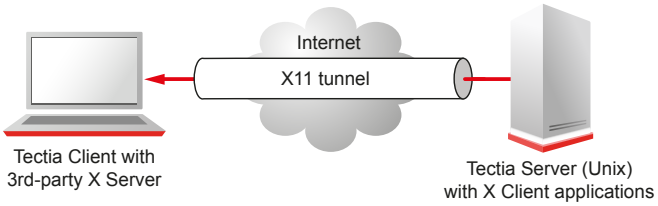


Figure 8.6. X11 forwarding

By default, Tectia Server allows X11 forwarding for all users. To enable X11 forwarding only for the specified users, include an entry similar to the following in your `ssh-server-config.xml` file:

```
<services>
  <rule group="admins">
    <tunnel-x11 action="allow" />
    ...
  </rule>
  <rule>
    <tunnel-x11 action="deny" />
  </rule>
</services>
```

On Unix, you can define what type of X11 listener address will be used in X11 forwarding. The address type is configured with the `settings` element by adding attribute `x11-listen-address` that takes the following values:

- `localhost` (default) - sets the DISPLAY environment variable to `127.0.0.1:<screen>`, where `<screen>` is the tunneled screen number, typically 10.0. This means that the x11 listener is bound to a loopback address; this setting should be sufficient for most use cases.

- `any` - sets the DISPLAY environment variable to `<address:screen>`, where `<address>` is the interface to which the SSH session is bound (typically the first network interface) and the `<screen>` is the tunneled screen number, typically 10.0. This setting will bind the X11 listener to the 0.0.0.0 (wildcard) interface thereby allowing connections to the proxy from other hosts. Use this setting on HPUX systems, if you need to tunnel older X11 applications (such as hpterm).

  When `x11-listen-address=any`, the SO_REUSEADDR socket option will be left non-set in order to prevent the possibility of session hijacking on some operating systems by other users binding to the same port with a more specific address.

For example:

```
<params>
  <settings
```

```
    x11-listen-address="any" />
</params>
```

8.5 Agent Forwarding (Unix)

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 host, and authentication passphrases or private keys never go over the network. For more information, see Section 5.13.

Tectia Client and ConnectSecure provide authentication agent functionality and the Connection Broker can also serve OpenSSH clients as an authentication agent. 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.

It is also possible to forward the certificate authentication data to obtain Kerberos ticket from a third host and continue authenticating to further hosts with the GSSAPI/Kerberos method.

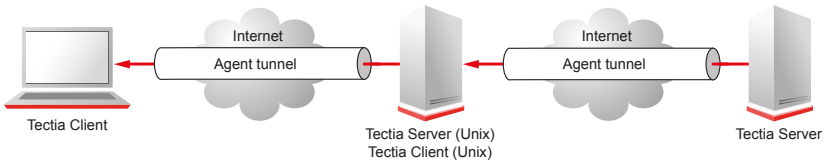


Figure 8.7. Agent forwarding

By default, Tectia Server allows agent forwarding for all users. To enable agent forwarding only for the specified users, include an entry similar to the following in your `ssh-server-config.xml` file:

```
<services>
  <rule group="admins">
    <tunnel-agent action="allow" />
    ...
  </rule>
  <rule>
    <tunnel-agent action="deny" />
  </rule>
</services>
```

Chapter 9 Troubleshooting Tectia Server

In case you encounter problems when running the Tectia Server software, you can try and solve the situation yourself first, and if that does not help, contact Tectia support.

Before you contact SSH technical support, run the **ssh-troubleshoot** (**ssh-troubleshoot.cmd** on Windows) tool to collect necessary information on your system and the installed 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 Tectia products are running. See instructions in Section 9.2.

For information on accessing the Tectia online support resources and contacting SSH technical support, see Section 1.2.

9.1 Starting Tectia Server in Debug Mode

You can run Tectia Server in debug mode to gather information which is useful when troubleshooting any connection or authentication problems.

Tectia Server control utility ssh-server-ctl(8) can be used to troubleshoot running service with **debug** command options that can be used to enable debug mode for the ssh-server-g3, servants and/or user SFTP server processes.

Note

Do not leave the server running in debug mode unnecessarily. Debugging slows down the performance of the server.

The following sections give instructions on activating the debugging on Unix and on Windows platforms.

9.1.1 Enabling Debug Mode for Running Tectia Server on Unix

To enable debug mode without restarting Tectia Server, run the following commands as privileged user:

1. Before enabling debug, check the status of the currently running server:

```
# ssh-server-ctl status -v
```

2. Log debug to a file, for example:

```
# ssh-server-ctl debug log-file /tmp/server-debug.txt
```

3. Enable debug mode. In this example, for all server processes with global debug level 4:

   ```
   # ssh-server-ctl debug set 4
   ```

   If higher `<debug_level>` is needed, it is recommended to use `"module=level,module*=level,..."` filter to enable debug only on intended modules. See next section for details on `filter`.

4. After reproducing the issue, remember to disable debug mode:

   ```
   # ssh-server-ctl debug clear
   ```

   This changes debug level to 0 and also removes the debug hook for the debug log file.

5. Check the status of the currently running server:

   ```
   # ssh-server-ctl status
   ```

   Verify from the output that 'Debug level' is 0. If Debug level is not displayed at all, the master ssh-server-g3 process in question has never had debug enabled.

Note

Stopping the service does NOT disable debug. If the service has already been restarted, then clear the debug settings also from the previous, retired service as it may still have retired server processes:

```
# ssh-server-ctl --old status
```

```
# ssh-server-ctl --old debug clear
```

6. To view all **debug** commands available for `ssh-server-ctl`:

   ```
   # ssh-server-ctl debug --help
   ```

9.1.2 Starting Tectia Server in Debug Mode on Unix

To start Tectia Server in debug mode, follow these instructions:

1. Stop the server, if it is currently running. Use the start-stop script described in Section 3.1:

2. Start the server executable by entering the following command with root privileges:

   ```
   # ssh-server-ctl start --server-log-file /tmp/server-debug.txt \
            --server-debug-level <filter>
   ```

Note

Recommended on Linux with systemd and on SELinux-enabled systems where manually starting the service is not supported.

Or manually, on other Unix systems:

```
# /opt/tectia/sbin/ssh-server-g3 -D<filter>
```

In the command:

- `filter` is an expression that takes the following syntax: `"module=level,module=level,..."` for example `"*Cert*=4,*Ocsp*=7"`. In case you define a filter with several modules with overlapping names, list the modules in increasing levels of detail (highest level last).
- `module` is an optional expression. It 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, for example `"2,SecShServer*=4"` or `"6,*Nio*=0"`
- `level` is an integer from 0 (no debug info) to 99 that specifies the desired amount of debug information.

Note

The levels 1-9 are the recommended debug levels. 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 server with a global debug level 4 and outputs the debug information to the screen:

```
# /opt/tectia/sbin/ssh-server-g3 -D4
```

The following example command starts the server listening on port 777, using only 1 servant process, and using filters to display level 2 output from everything else, but level 4 output from modules starting with "SecShServer". The debug output is redirected to file `server.log`.

```
# /opt/tectia/sbin/ssh-server-g3 -l 777 -n 1 -D"2,SecShServer*=4" > server.log 2>&1
```

To view all options available for `ssh-server-g3`, enter the following command:

```
# ssh-server-g3 -h
```

9.1.3 Enabling Debug Mode for Running Tectia Server on Windows

To enable debug mode without restarting Tectia Server, run the following commands as privileged user:

1. Start *Windows PowerShell* with *Run as Administrator* and change to the "`<INSTALLDIR>\SSH Tectia Server\`", for example:

   ```
   cd "C:\Program Files (x86)\SSH Communications Security\SSH Tectia\SSH Tectia Server\"
   ```

2. Before enabling debug, check the status of the currently running server:

   ```
   .\ssh-server-ctl status -v
   ```

3. Log debug to a file, for example:

   ```
   .\ssh-server-ctl debug log-file C:\server-debug.txt
   ```

Or alternatively, View Troubleshooting Log if you wish to both monitor and store the log contents to a file simultaneously:

```
.\ssh-server-debugger
```

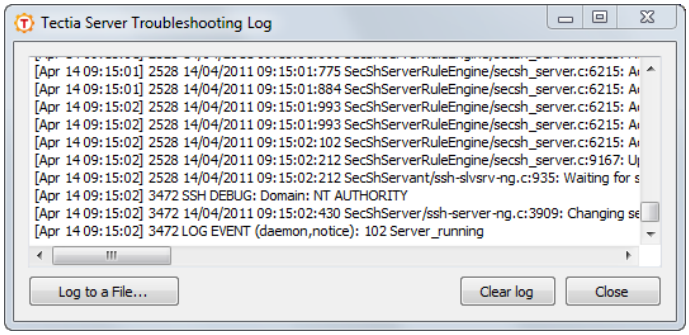


Figure 9.1. Viewing troubleshooting log

4. Enable debug mode. In this example, for all server processes set global debug level 4 and modules starting with "SshUser" higher level 9:

   ```
   .\ssh-server-ctl debug set "4,SshUser*=9"
   ```

   This `<debug_level>` is suitable for user authentication. The amount of debug can also be reduced by disabling some of the other modules by setting them to 0 in `"module=level,module*=level,..."`. See previous section for details on `filter`.

5. After reproducing the issue, remember to disable debug mode:

   ```
   .\ssh-server-ctl debug clear
   ```

   This changes debug level to 0 and also removes the debug hook for the debug log file if it was used.

6. Check the status of the currently running server:

   ```
   .\ssh-server-ctl status
   ```

   Verify from the output that 'Debug level' is 0. If Debug level is not displayed at all, the master ssh-server-g3 process in question has never had debug enabled.

7. To view all **debug** commands available for **ssh-server-ctl** see:

   ```
   .\ssh-server-ctl debug --help
   ```

9.1.4 Starting Tectia Server in Debug Mode on Windows

Note

On Windows, to get debug output from the Windows server using Tectia Server Configuration GUI, you need to have appropriate Windows policy rights to have the "Debug programs" policy enabled. The admin user has the policy enabled by default.

To start the Tectia Server in debug mode, follow these instructions:

1. Open the **Tectia Server Configuration** tool from the Start menu by selecting: **Start → Program Files (x86) → SSH Tectia Server → SSH Tectia Server Configuration**

   The following window appears:

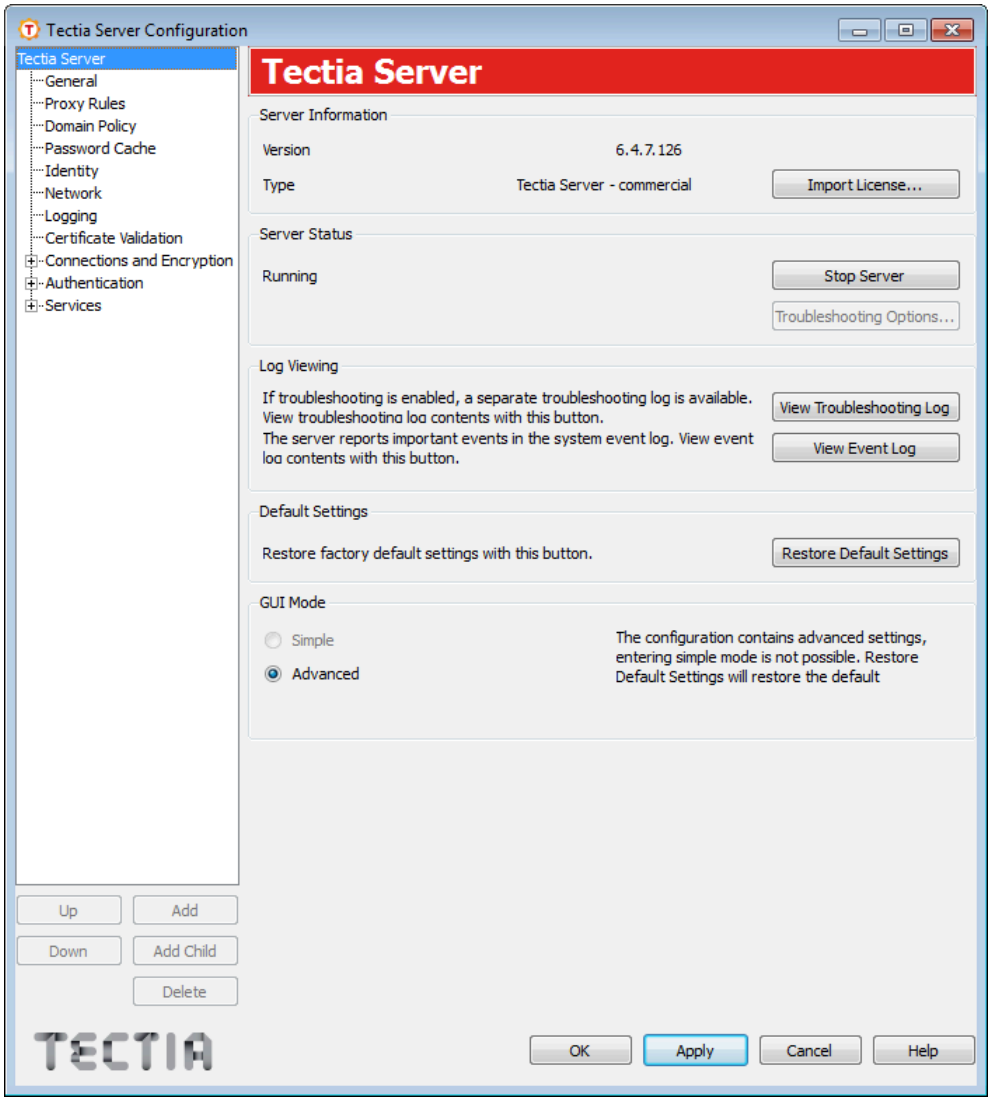


Figure 9.2. Tectia Server Configuration - Tectia Server page

2. Stop the server, if it is currently running, by clicking the **Stop Server** button. This will enable the **Troubleshooting Options** button.

3. Click the **Troubleshooting Options** button and select the **Run server in troubleshooting mode** option. Additionally, you can go under **Troubleshooting string** and specify the level of information to display. The string can be a `filter` or a number from 1 to 99 (the default is 2). The higher the number, the more detailed the troubleshooting output will be, and the more the debugging will affect performance.

Note

The levels 1-9 are the recommended debug levels. Important: if `<debug_level>` above 4 is needed, it is recommended to use a `filter` instead of enabling global level for all modules. For example `"4,SshUser*=9"` or `"*Cert*=4,*Ocsp*=7"` can be used to limit debug only to specific modules for user authentication.

In case you define a filter with several modules with overlapping names, list the modules in increasing levels of detail (highest level last).

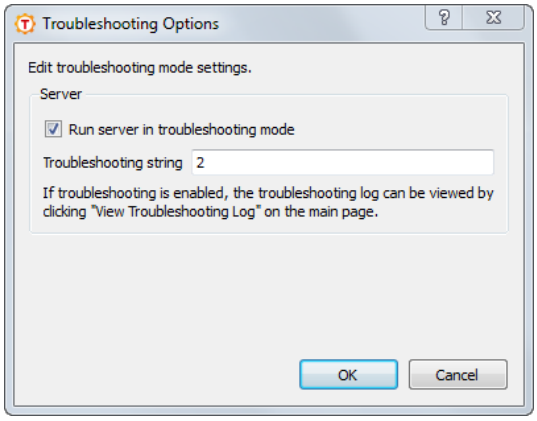


Figure 9.3. Editing troubleshooting options

4. Click **OK** to close the Troubleshooting Options window.

5. Click the **View Troubleshooting Log** button to open a window where the troubleshooting information will be displayed.

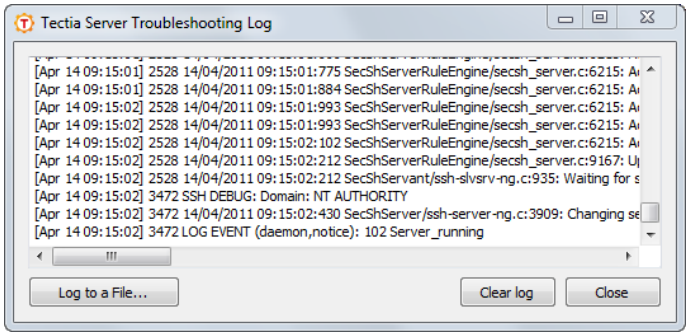


Figure 9.4. Viewing troubleshooting log

You can store the log contents into a file by clicking the **Log to a File** button. Define the file name and location as applicable.

6. Start the server by clicking the **Start Server** button.

The troubleshooting information will now be displayed in a window named **Tectia Server Troubleshooting Log**.

After troubleshooting the server follow these steps to disable the troubleshooting:

1. Stop the server by clicking the **Stop Server** button.
2. Click **Troubleshooting Options** and unselect the **Run server in troubleshooting mode** option.
3. Start the server by clicking the **Start Server** button.

9.1.5 Debugging Secure File Transfer

The secure file transfer process (**sft-server-g3**) is debugged together with the Tectia Server, and the same debug level that is set for Tectia Server (**ssh-server-g3**) is used for both processes by default.

Tectia Server control utility **ssh-server-ctl** can be used to troubleshoot running service with **debug** command. There are file transfer related options that can be used to enable debug mode for the user SFTP server processes without the need to set any environment variables.

```
ssh-server-ctl debug set "4;sftpfile=/tmp/sftp_debug_%U.txt"
```

This will create user-specific SFTP debug logs on the server-side. There are two SFTP related debug level options `sftpfile=<filename>` and `sftpdebug=<level>`. If only `sftpdebug` is defined, the debug messages are sent to the ssh-servant-g3 server process.

For more information on SFTP debug options, please see the ssh-server-ctl ssh-server-ctl(8) and ssh-server-config(5) man pages.

Alternatively, the environmental variables can be used to redirect the **sft-server-g3** debug messages, that are by default sent to the standard error, that goes also to the SFTP client. If you want to forward the **sft-server-g3** debug messages into a file and not to the client, you can add the following two environment variables affecting the secure file transfer user into the `/etc/environment` file on Unix and into the user-specific environment variables on Windows:

1. **SSH_SFTP_DEBUG** defines the debug level that controls the messages that the **sft-server-g3** process will be showing while executing. This variable can be used when no debug level has been set for the **sft-server-g3** process. The value can be for example:

   ```
   SSH_SFTP_DEBUG=SftpLibServer=80,SftpLibStageFsFile=80
   ```

   If debug level is set for **ssh-server-g3** this variable will be ignored.

2. **SSH_SFTP_DEBUG_FILE** defines the file where the debug messages from the **sft-server-g3** will be printed. This variable can be used when no debug level has been set for the **sft-server-g3** process. The value can be for example:

   ```
   SSH_SFTP_DEBUG_FILE=/tmp/sft_debug.txt
   ```

   If this variable is not defined, the messages will be sent to standard error as normally, and the SFTP client will receive them.

If debug level is set for **ssh-server-g3** this variable will be ignored.

9.2 Collecting System Information for Troubleshooting

Tectia Server includes a troubleshooting tool that automatically collects necessary data about the operating system and hardware, and about the installed 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
- Tectia binaries, the tool checks the actual installation package versions and detects also debug packages
- Tectia global configuration from the `/etc/` and `/opt/` directories on Unix, and from the default installation directory on Windows:
  - `"C:\Program Files (x86)\SSH Communications Security\SSH Tectia"` on 64-bit Windows versions
- User-specific 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.

To collect system information, open a command prompt and enter the following command:

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(8).

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 Technical Support for analysis.

Caution

Handle the output file with appropriate care as it may contain security-critical data.

9.3 Solving Problem Situations

The following sections give workaround instructions for a few problem situations that may occur with Tectia Server:

- For CPU overload on HP-UX, see Section 9.3.1
- For server failing to start because of wrong host key permissions on Windows, see Section 9.3.2
- For server failing to start because of wrong configuration file permissions on Windows, see Section 9.3.3
- For domain account authentication issues on Windows, see Section 9.3.4
- For incorrect login times on Windows, see Section 9.3.5
- For failing access to Windows domain resources, see Section 9.3.6

9.3.1 CPU Overload on Tectia Server on HP-UX

Symptom: Tectia Server installed on HP-UX 11.11 takes a lot of CPU, maybe upto 100%.

Check that you have installed the latest versions of the patches offered by HP. At least the following Kerberos-related patches are needed for the PAM Kerberos to operate properly:

- PHCO_34214, libpam_unix cumulative patch, July 2006
- PHNE_25779, LDAP-UX Integration B.02.00 cumulative patch, March 2002
- PHSS_33384, KRB5-Client Version 1.0 cumulative patch, June 2005
- PHSS_35381, s700_800 11.11 ld(1) and linker tools cumulative patch, February 2007

Check the Hewlett-Packard web site for the latest versions.

9.3.2 Invalid Host Key Permissions on Windows

Symptom: Tectia Server fails to start and reports error "Invalid hostkey permissions for hostkey". This occurs usually after restroring backups or after upgrading Tectia Server from 4.x to 6.x.

The permissions of the server host key file and directory have been made more strict since the 4.x releases. In 6.x, full permissions are allowed only for the Administrators group and the SYSTEM account, and no other permissions are set at all.

The host key permissions can be updated manually by using the **ssh-keygen-g3** tool:

1. Go to the Tectia Server installation directory:

   - `"C:\Program Files (x86)\SSH Communications Security\SSH Tectia"` on 64-bit Windows versions

2. Set the permissions for the host key by running command:

```
$ ssh-keygen-g3 --set-hostkey-owner-and-dacl hostkey
```

For more information on the tool, see ssh-keygen-g3(1).

9.3.3 Invalid Configuration File Permissions on Windows

Symptom: Tectia Server fails to start and logs "Error in initial configuration".

Make sure that the permissions of the server configuration file `ssh-server-config.xml` are as follows:

- The owner of the file is a member of the *Administrators* group.
- Only *Administrators* and *SYSTEM* may have **Full control** of the file.
- *Users* have at most **Read & execute** permissions - they are not allowed to modify the file.
- Other accounts do not have access to the file.

9.3.4 Authentication Fails for Domain Account on Tectia Server on Windows

Symptom: User authentication fails for domain accounts on Tectia Server.

This problem could be caused by a number of reasons:

Start the troubleshooting by checking that you are able to log on to the Windows host where Tectia Server is installed. Attempt the logon at the console using domain accounts. Additionally, check that the Windows host where Tectia Server is installed can reach the Domain Controller. While Windows allows you to log on (by using cached passwords) at the console even when the Domain Controller is down, Tectia Server requires active communication with the Domain Controller in order to authenticate domain users.

Another possible cause for this is having additional groups in the "Pre-Windows 2000 Compatible Group" on the Domain Controller. Some Windows updates, such as the multilanguage pack, can result in adding entries to this group, which breaks the mapping for the groups' SIDs, and Tectia Server is no longer able to create access tokens for the domain users. In order to prevent this, check on the Domain Controller that the "Pre-Windows 2000 Compatible Group" contains only the "Authenticated Users" group.

9.3.5 Last Login Time is Incorrect on Windows

Symptom: After a successful user authentication, Tectia Server displays the previous time when the same user logged in to this server. However, in some cases the last login time shown is incorrect.

The last login time may differ from the server's time because the timestamp comes from the domain controller (DC), not from the host where Tectia Server is running. In large environments with multiple DCs, the timestamps are not replicated between DCs. Logging in using one DC will not update the timestamps shown by other DCs. The last login time may be unreliable if the time on the DCs is not set accurately.

9.3.6 Virtual Folders Defined on Windows Network Shared Folders Are Not Available on Tectia Server on Windows

Symptom: Users cannot see or access their virtual folders defined to point to Windows network shared folder on Tectia Server running on a Windows platform.

Folders on Windows network shares are accessible with password authentication on all supported Windows platforms as long as the Windows interactive logon is used. Ensure the domain user has 'log on locally' access right and Tectia Server configuration has **windows-logon-type** set as interactive.

With other authentication methods (such as public keys, GSSAPI, or certificates), access to Windows network shares can be enabled only when combined with Tectia Server **password-cache** feature or in older native domains with functional level of Windows Server 2003 when the following requirements are met (or if these methods are used together with password authentication, see Section 5.17):

- The Kerberos extension S4U is applied
- The delegation is set correctly on the Domain Controller

Follow these instructions to set up the delegation in the Active Directory:

1. Log in to the Domain Controller.
2. Open the **Active Directory Users and Computers** snap-in

   OR open the corresponding tool in **Start→Programs→Administrative Tools**.
3. Open the **Computers** tree and select the computer where the Tectia Server is located.
4. Right-click and select **Properties**.
5. Select the Delegation tab and make the following settings:
   a. Select **Trust this computer for delegation to specified services only**.
   b. Select **Use any authentication protocol**.
   c. Click the **Add** button.

d. Click the **Users or Computers** button.

e. Enter the name of the host where the network share is located and click **Ok**.

f. Select **cifs** (common internet file system) from the available services.

6. Click **Ok** to close the open windows.

Appendix A Tectia Server Configuration File Quick Reference

This Appendix contains a quick reference to the elements of the Tectia Server configuration file, `ssh-server-config.xml`. The quick reference is divided into four tables, one for each block of the configuration file:

- Table A.1: The `params` block (General server parameters)
- Table A.2: The `connections` block (Connection rules and encryption methods)
- Table A.3: The `authentication-methods` block (Authentication rules and methods)
- Table A.4: The `services` block (Service rules)

The tables list the available configuration file elements with their attributes, attribute values (with the default value, if available, marked in bold) and descriptions. The element names in the tables are links that take you to detailed descriptions of the elements in ssh-server-config(5).

The element hierarchy is expressed with slashes ('/') between parent and child elements. For example, in Table A.2 , "`connection / selector / ip`" means that a `connection` element can have a `selector` child element, which can have an `ip` child element.

Table A.1. ssh-server-config.xml Quick Reference - the `params` block

Element	Attributes and their values	Description
address-family	`type = "`**`inet`**`|inet6|any"`	IP address type
crypto-lib	`mode = "`**`standard`**`|fips"`	Cryptographic library mode
settings	`proxy-scheme`	HTTP and SOCKS proxy server rules for local tunneling

Element	Attributes and their values	Description
	`= semicolon-separated_sequence`	
	`xauth-path = path` *(Unix only)*	Path to a supplementary XAuth binary used with X11 forwarding
	`xauth-shell = shell` *(Unix only)*	Shell used to run the XAuth binary. Default is the user shell.
	`x11-listen-address = "localhost|any"` *(Unix only)*	Type of address the x11 listener is created on
	`pam-account-checking-only = "yes|no"` *(Unix only)*	Only PAM will be used to check if the user is allowed to log in
	`resolve-clienthostname = "yes|no"`	Client host name is resolved from IP address during connection setup
	`ignore-aix-rlogin = "yes|no"`	Ignore remote login restriction on AIX
	`ignore-aix-login = "yes|no"`	Ignore local login restriction on AIX
	`record-ptyless-sessions = "yes|no"`	Record sessions without PTYs as user logins in the OS
	`user-config-dir = directory` (default: `"%D/.ssh2"`)	Directory for user-specific configuration data (can include pattern strings)
	`default-path = path` *(Unix only)*	Default PATH value for the user environment
	`windows-logon-type = "batch|interactive|network|network-cleartext"` *(Windows only)*	Accepted user logon methods for the local host
	`windows-terminal-mode = "console|stream"` *(Windows only)*	Mode of operation of a terminal session on the server side
	`ignore-nisplus-no-permission = "yes|no"` *(Linux and Solaris only)*	If NIS+ gives no permission to the user during authentication, ignore it
	`quiet-login = "yes|no"`	Suppress messages about last login, password expiry, etc. during login
	`default-domain = domain`	Append a domain to server host names that are not FQDNs
`terminate-user-processes = "yes|no"`	Terminate user processes on session close	

Element	Attributes and their values	Description
`allow-elevation = "yes|no"`	Allow elevation. Only applies to password logins.	
pluggable-authentication-modules *(Unix only)*	`pam-calls-with-commands = "yes|no"`	Enable PAM Account and Session Management when user executes shells, remote commands and subsystems
	`service-name =` *`name`*	Instruct PAM about which configuration it should use
	`dll-path =` *`path`*	Location of the PAM library
protocol-parameters	`threads =` *`number`* (default: "**0**")	Number of threads the protocol library uses
hostkey / **private**	`file =` *`path`*	Path to the private key file
hostkey / **public**	`file =` *`path`*	Path to the public key file
hostkey / **x509-certificate**	`file =` *`path`*	Path to the X.509 host certificate file
hostkey / **openssh-certificate**	`file =` *`path`*	Path to the OpenSSH host certificate file
hostkey / **externalkey**	`type = "none|software|mscapi|pkcs11|pkcs12"`	External host key type
	`init-info =` *`keyword(value)_list`*	Init info for the external host key
listener	`id =` *`ID`*	Unique ID for the server listener
	`address =` *`IP_address`*	The address where the server listens for connections
	`port =` *`port_number`*	The port at which the server listens for connections
domain-policy *(Windows only)*	`windows-domain-precedence =` *`comma-separated_list`*	Trusted domains and special values `%default%` and `%local%`
domain-policy / **windows-domain** *(Windows only)*	`name =` *`domain_name`*	Domain name for domain access with one-way trust
	`user =` *`user_name`*	User account for domain access with one-way trust
logging / **log-events**	`facility = "normal|daemon|user|auth|local0|local1|local2|local3|local4|local5|local6|local7|discard"`	Facility of logging event
	`severity = "informational|notice|warning|error|critical|security-success|security-failure"`	Severity of logging event

Element	Attributes and their values	Description
limits	`max-processes = [1 to 2048]` (default: "**40**")	Maximum number of servant processes the master server will launch
	`max-connections = number` (default: "**256**")	Maximum number of client connections allowed per servant
limits / **servant-lifetime**	`total-connections = [1 to 4000000000]` (recommended: "5000")	Total number of connections the servant process will handle during its lifetime
cert-validation	`http-proxy-url = address`	HTTP proxy address
	`socks-server-url = address`	SOCKS proxy address
	`cache-size = [1 to 512]` (default: "**300**")	Maximum size (MB) of in-memory cache for certificates and CRLs
	`max-crl-size = [1 to 512]` (default: "**50**")	Maximum size (MB) of CRLs accepted
	`external-search-timeout = [1 to 3600]` (default: "**60**")	Time limit (seconds) for external HTTP and LDAP searches for CRLs and certificates
	`max-ldap-response-length = [1 to 512]` (default: "**50**")	Maximum size (MB) of LDAP responses accepted
	`ldap-idle-timeout = [1 to 3600]` (default: "**30**")	Idle timeout (seconds) for LDAP connections
	`max-path-length = number`	Maximum length of the certification paths when validating certificates
cert-validation / **ldap-server**	`address = LDAP-address`	LDAP server address
	`port = port_number` (default: "**389**")	LDAP server port
cert-validation / **ocsp-responder**	`validity-period = seconds`	Validity period for OCSP data
	`url = address`	OCSP responder service address
cert-validation / **cert-cache-file**	`file = path`	File for storing certificates and CRLs
cert-validation / **crl-auto-update**	`update-before = seconds`	Time before expiration for automatic updating of certificate revocation lists
	`minimum-interval = seconds`	Limit for maximum CRL update frequency
cert-validation / **crl-prefetch**	`url = address`	URL from which CRL is downloaded
	`interval = seconds` (default: "**3600**")	How often the CRL is downloaded

Element	Attributes and their values	Description
cert-validation / **dod-pki**	`enable = "yes|no"`	Enforce digital signature in key usage
cert-validation / **ca-certificate**	`name = CA_name`	Name of the CA
	`file = path`	Path to X.509 CA certificate file
	`disable-crls = "yes|no"`	Disable CRL checking
	`use-expired-crls = seconds` (default: "**0**")	Time period for using expired CRLs
	`trusted = "yes|no"`	Set CA certificate as a trust anchor and trust it explicitly
cert-validation / **openssh-ca-key**	`name = OpenSSH_CA_name`	Name of the OpenSSH certification authority (CA) used in server authentication
	`file = path`	Path to the OpenSSH CA public key file
password-cache	`file = path`	Location of server password cache file
load-control	`enable = "yes|no"`	Enable load control
	`discard-limit = [1 to max-connections-1]` (default: **90% of max-connections**)	Limit for discarding new connections from outside the server's white list
	`white-list-size = [1 to 10000]` (default: "**1000**")	Number of IP addresses on the server's white list

Table A.2. ssh-server-config.xml Quick Reference - the `connections` block

Element	Attributes and their values	Description
connection	`name = XML_name`	Identifier (valid XML name) for the connection rule
	`action = "allow|deny"`	Allow/deny connection
	`tcp-keepalive = "yes|no"`	Send keepalive messages to the other side
connection / **selector** / **interface**	`id = ID`	Match the server listener interface ID
	`address = address`	Match the server listener interface address
	`port = port_number`	Match the server listener interface port
connection / **selector** / **ip**	`address = IP_address |IP_address_range |IP_sub-network_mask`	Match the client's IP address
	`fqdn = FQDN_pattern`	Match the client's FQDN

Element	Attributes and their values	Description
connection / **rekey**	`seconds = seconds` (default: "**3600**")	Number of seconds after which key exchange is done again
	`bytes = bytes` (default: "**1000000000**")	Number of transferred bytes after which key exchange is done again
connection / **cipher**	`name = cipher_name`	Cipher allowed for data encryption
	`allow-missing = "yes|no"`	Server restarts normally even if cipher not found during configuration reading
connection / **mac**	`name = HMAC_name`	MAC allowed for data integrity verification
	`allow-missing = "yes|no"`	Server restarts normally even if MAC not found during configuration reading
connection / **kex**	`name = KEX_name`	KEX allowed for key exchange method
	`allow-missing = "yes|no"`	Server restarts normally even if KEX not found during configuration reading
connection / **hostkey-algorithm**	`name = algorithm_name`	Host key signature algorithm used in server authentication with host keys or certificates
	`allow-missing = "yes|no"`	Server restarts normally even if host key algorithm not found during configuration reading

Table A.3. ssh-server-config.xml Quick Reference - the `authentication-methods` block

Element	Attributes and their values	Description
banner-message	`file = path`	Path to the file that contains the message that is sent to the client before authentication
auth-file-modes *(Unix only)*	`strict = "yes|no"`	Check permissions and ownership of the user's key files or the directory they are stored in
	`mask-bits = octal_permissions` (default: "**022**")	Specify forbidden permission bits in octal format
	`dir-mask-bits = octal_permissions`	Specify the forbidden permission bits for the user key directory
authentication	`action = "allow|deny"`	Allow/deny access to/from users who match a selector

Element	Attributes and their values	Description
authentication / **selector** / **certificate**	`field = "ca-list|issuer-name |subject-name|serial-number |altname-email|altname-upn |altname-ip|altname-fqdn |extended-key-usage"`	The field of user certificates used in public-key authentication that has to be matched
	`pattern`	The information in the `field` to be matched
	`pattern-case-sensitive`	The information in the `field` to be matched case-sensitively
	`regexp =` *`egrep_regexp`*	Regular expression to match a range of values in the selected field
	`ignore-prefix = "yes|`**`no`**`"`	Match only the end of subject name
	`ignore-suffix = "yes|`**`no`**`"`	Match only the beginning of the subject name
	`explicit = "yes|no"`	(With `extended-key-usage`) Request that the certificate must include the key purpose ID specified with the pattern
	`allow-undefined = "yes|`**`no`**`"`	Control behavior of selector when required data is not defined
authentication / **selector** / **host-certificate**	`field = "ca-list|issuer-name |subject-name|serial-number |altname-email|altname-upn |altname-ip|altname-fqdn |extended-key-usage"`	The field of host certificates used in public-key authentication that has to be matched
	`pattern`	The information in the `field` to be matched
	`pattern-case-sensitive`	The information in the `field` to be matched case-sensitively
	`regexp =` *`egrep_regexp`*	Regular expression to match a range of values in the selected field
	`ignore-prefix = "yes|`**`no`**`"`	Match only the end of subject name
	`ignore-suffix = "yes|`**`no`**`"`	Match only the beginning of the subject name
	`explicit = "yes|no"`	(With `extended-key-usage`) Request that the certificate must include the key purpose ID specified with the pattern
	`allow-undefined = "yes|`**`no`**`"`	Control behavior of selector when required data is not defined
authentication / **selector** / **interface**	`id =` *`ID`*	Match the listener interface ID
	`address =` *`IP_address`*	Match the listener address

Element	Attributes and their values	Description
	`port = port_number`	Match the listener port
	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication / **selector** / **ip**	`address = IP_address |IP_address_range |IP_sub-network_mask`	Match client's IP address
	`fqdn = FQDN_pattern`	Match client's FQDN
	`fqdn-regexp = regexp_pattern`	Match a range of FQDNs specified with a regular expression
	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication / **selector** / **user**	`name = comma-separated_list`	Match user names
	`name-case-sensitive = comma-separated_list`	Match user names case-sensitively
	`name-regexp = regexp_pattern`	Match a range of names specified with a regular expression
	`id = comma-separated_list`	Match user IDs
	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication / **selector** / **user-group**	`name = comma-separated_list`	Match user group names
	`name-case-sensitive = comma-separated_list`	Match user group names case-sensitively
	`name-regexp = regexp_pattern`	Match a range of user group names specified with a regular expression
	`id = comma-separated_list`	Match user group IDs
	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication / **selector** / **user-privileged**	`value = "yes|no"`	Match a privileged user
	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication / **selector** / **blackboard**	`field`	Match based on the information in this blackboard field
	`pattern`	The information in the `field` to be matched
	`pattern-case-sensitive`	The information to be matched case-sensitively
	`regexp = egrep_regexp`	Regular expression to match a range of values in the selected field
	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication /	`length = [ length_range ]`	Public key length range

Element	Attributes and their values	Description
selector / **publickey-passed**	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication / **selector** / **user-password-change-needed** *(Unix only)*	`value = "yes|no"`	Matches if the user password has expired and should be changed
	`allow-undefined = "yes|no"`	Control behavior of selector when required data is not defined
authentication / **set-blackboard**	`field = blackboard_key`	Describe an item that will be added to the blackboard when this authentication block is encountered
	`value`	Desired value
	`file = path`	Path to a file containing the desired value
authentication / **set-user**	`name = user_name`	Specify user name that will be used from here on
authentication / **auth-publickey**	`require-dns-match = "yes|no"`	Accept or deny a public key which has the `allow/deny-from` option set in the authorization file
	`signature-algorithms = comma-separated_list`	Public-key signature algorithms used for user authentication
	`authorization-file = comma-separated_list`	Paths to files that contain the user public keys that are authorized for login
	`authorized-keys-directory = comma-separated_list`	Directories that contain the user public keys that are authorized for login
	`openssh-authorized-keys-file = comma-separated_list`	Paths to OpenSSH-style authorized_keys files that contain the user public keys that are authorized for login
authentication / **auth-hostbased**	`require-dns-match = "yes|no"`	Host-based authentication will require the host name given by the client to match the one found in DNS
	`disable-authorization = "yes|no"`	Host-based authentication ignores authorization requirements
	`allow-missing = "yes|no"`	Ignore missing element
authentication / **auth-password**	`failure-delay = seconds` (default: "2")	Delay between failed password authentication attempts
	`max-tries = number` (default: "3")	Maximum number of password authentication attempts
	`allow-missing = "yes|no"`	Ignore missing element

Element	Attributes and their values	Description
authentication / **auth-keyboard-interactive**	`failure-delay = seconds` (default: "**2**")	Delay between failed keyboard-interactive authentication attempts
	`max-tries = number` (default: "**3**")	Maximum number of keyboard-interactive authentication attempts
authentication / **auth-keyboard-interactive** / **submethod-pam** *(Unix only)*	`service-name`	Instruct PAM about which configuration it should use
	`dll-path = path |comma-separated_list`	Non-standard location for the PAM library, or PAM DLLs
authentication / **auth-keyboard-interactive** / **submethod-password**	-	Set the keyboard-interactive password submethod in use
authentication / **auth-keyboard-interactive** / **submethod-securid**	`dll-path = path`	Path to the SecurID DLL
authentication / **auth-keyboard-interactive** / **submethod-radius**	-	Sets the keyboard-interactive RADIUS submethod in use
authentication / **auth-keyboard-interactive** / **submethod-radius** / **radius-server**	`address = IP_address`	RADIUS server's IP address
	`port = port_number` (default: "**1812**")	RADIUS server port
	`timeout = seconds` (default: "**10**")	Time after which the RADIUS query is terminated if no response is gained
	`client-nas-identifier = ID`	Network access server identifier to be used when talking to the RADIUS server
authentication / **auth-keyboard-interactive** / **submethod-radius** / **radius-server** / **radius-shared-secret**	`file = path`	Path to the RADIUS shared secret file
authentication / **auth-keyboard-interactive** / **submethod-aix-lam**	`enable-password-change = "yes|no"`	Enable LAM on AIX and allow users to change their expired passwords
authentication /	`name = method_name`	Set the named generic submethod in use

Element	Attributes and their values	Description
auth-keyboard-interactive / **submethod-generic**	`params = parameters`	Optional parameters for the submethod
authentication / **auth-gssapi**	`dll-path = path`	Path to required GSSAPI libraries
	`allow-ticket-forwarding = "yes|no"`	Allow forwarding the Kerberos ticket over several connections
	`allow-missing = "yes|no"`	Ignore Kerberos/GSSAPI unavailability
authentication / **mapper**	`command = external_application`	External application used to supplement authentication
	`timeout = [1 to 3600]` (default: "**15**")	Time limit for the external application to exit

Table A.4. ssh-server-config.xml Quick Reference - the `services` block

Element	Attributes and their values	Description
group	`name = XML_name`	Group name (a valid XML name)
group / **selector**	This element has the same child elements as `authentication-methods / authentication / selector` (see Table A.3)	
rule	`group = group_name`	Match user's group
	`idle-timeout = seconds` (default: "**0**")	Idle timeout limit
	`print-motd = "yes|no"`	Print message of the day at interactive login to a Unix server
rule / **environment**	`allowed = comma-separated_list`	Environment variables the user group is allowed to set at the client side
	`allowed-case-sensitive = comma-separated_list`	Specify case-sensitive variables
rule / **terminal**	`action = "allow|deny"`	Allow/deny terminal access for the user group
	`chroot = directory` *(Unix only)*	Directory where user is chrooted during the terminal session
rule / **subsystem**	`type = subsystem`	Subsystem for which the settings are made
	`action = "allow|deny"`	Allow/deny use of the subsystem
	`audit = "yes|no"`	Record audit messages of the subsystem in the system log
	`exec-directly = "yes|no"` (*Unix only*)	Server will launch **sft-server-g3** directly without invoking the user's shell
	`application = executable`	The executable of the subsystem

Element	Attributes and their values	Description
	`chroot = directory`	Directory where the user is chrooted when running the subsystem
rule / **subsystem** / **attribute**	`name = attribute_name`	Name for the subsystem attribute
	`value = attribute_value`	Value of the subsystem attribute
rule / **command**	`action = "allow|deny|forced"`	Allow/deny/force shell command
	`interactive = "yes|no"` *(Windows only)*	For `forced` action: the `application` requires user interaction
	`application = application_name`	The application that is allowed/ forced to run
	`application-case-sensitive = application_name`	(Alternative to `application`:) The application is matched case-sensitively
	`chroot = directory`	Directory where user is chrooted when running the command
rule / **tunnel-agent**	`action = "allow|deny"`	Allow/deny agent forwarding
rule / **tunnel-x11**	`action = "allow|deny"`	Allow/deny X11 forwarding
rule / **tunnel-local**	`action = "allow|deny"`	Allow/deny local tunnels
rule / **tunnel-local** / **src**	`address = IP_address |IP_address_range |IP_sub-network_mask`	Source address for client using the local tunnel
	`fqdn = FQDN_pattern`	Source FQDN for client (matches case-insensitively)
	`fqdn-regexp = regexp_pattern`	Regular expression (egrep) to match a range of FQDNs
rule / **tunnel-local** / **tunnel-src**	`address = IP_address |IP_address_range |IP_sub-network_mask`	Source address for local tunnel
	`fqdn = FQDN_pattern`	Source FQDN for local tunnel (matches case-insensitively)
	`fqdn-regexp = regexp_pattern`	Regular expression (egrep) to match a range of FQDNs
rule / **tunnel-local** / **dst**	`address = IP_address |IP_address_range |IP_sub-network_mask`	Destination address for local tunnel
	`fqdn = FQDN_pattern`	Destination FQDN for local tunnel (matches case-insensitively)
	`fqdn-regexp = regexp_pattern`	Regular expression (egrep) to match a range of FQDNs
	`port = port_number`	Destination port or port range for local tunnel

Element	Attributes and their values	Description
rule / **tunnel-local** / **mapper**	`command =` *`external_application`*	External application which is the executable of the subsystem
	`timeout = [1 to 3600]` (default: "**15**")	Time limit for the external application to exit
rule / **tunnel-remote**	`action = "`**`allow`**`|deny"`	Allow/deny remote tunnels
rule / **tunnel-remote** / **src**	`address =` *`IP_address`* `|`*`IP_address_range`* `|`*`IP_sub-network_mask`*	Source address for remote tunnel
	`fqdn =` *`FQDN_pattern`*	Source FQDN for remote tunnel (matches case-insensitively)
	`fqdn-regexp =` *`regexp_pattern`*	Regular expression (egrep) to match a range of FQDNs
rule / **tunnel-remote** / **tunnel-dst**	`address =` *`IP_address`* `|`*`IP_address_range`* `|`*`IP_sub-network_mask`*	Destination address for remote tunnel
	`fqdn =` *`FQDN_pattern`*	Destination FQDN for remote tunnel (matches case-insensitively)
	`fqdn-regexp =` *`regexp_pattern`*	Regular expression (egrep) to match a range of FQDNs
rule / **tunnel-remote** / **listen**	`address =` *`IP_address`* `|`*`IP_address_range`* `|`*`IP_sub-network_mask`*	Listen address for remote tunnel
	`port =` *`port_number`*	Listen port or port range for remote tunnel

Appendix B Server Configuration File Syntax

The DTD of the server configuration file is shown below:

```
<!--                                                                    -->
<!--                                                                    -->
<!--                                                                    -->
<!-- secsh-server.dtd                                                   -->
<!--                                                                    -->
<!-- Copyright (c) 2022 SSH Communications Security Corporation.        -->
<!-- This software is protected by international copyright laws.        -->
<!-- All rights reserved.                                               -->
<!--                                                                    -->
<!-- Document type definition for the Tectia Server XML                 -->
<!-- configuration files.                                               -->
<!--                                                                    -->
<!--                                                                    -->

<!-- Tunable parameters used in the policy. -->

<!-- Default connection action. -->
<!ENTITY default-connection-action                    "allow">

<!-- Default terminal action. -->
<!ENTITY default-terminal-action                      "allow">

<!-- Default subsystem action. -->
<!ENTITY default-subsystem-action                     "allow">

<!-- Default subsystem audit value. -->
<!ENTITY default-subsystem-audit                      "yes">
```

```
<!-- Default for allowing undefined blackboard entries by selectors. -->
<!ENTITY default-allow-undefined-value                  "no">

<!-- Default user-privileged value. -->
<!ENTITY default-user-privileged-value                  "yes">

<!-- Default user-password-change-needed value. -->
<!ENTITY default-user-password-change-needed-value      "yes">

<!-- Reverse mapping is not required by default in
     publickey authentication. -->
<!ENTITY default-auth-publickey-require-dns-match       "no">

<!-- Default tunnel action. -->
<!ENTITY default-tunnel-action                          "allow">

<!-- Default command action. -->
<!ENTITY default-command-action                         "allow">

<!-- Default interactive command action. -->
<!ENTITY default-interactive-command-action             "no">

<!-- Default rekey interval in seconds. -->
<!ENTITY default-rekey-interval-seconds                 "3600">

<!-- Default rekey interval in bytes (1GB). -->
<!ENTITY default-rekey-interval-bytes                   "1000000000">

<!-- Default login grace time in seconds. -->
<!ENTITY default-login-grace-time-seconds               "600">

<!-- Default authentication action. -->
<!ENTITY default-authentication-action                  "allow">

<!-- Password authentication default failure delay in seconds. -->
<!ENTITY default-auth-password-failure-delay            "2">

<!-- Password authentication default maximum tries. -->
<!ENTITY default-auth-password-max-tries                "3">

<!-- Password cache is disabled by default -->
<!ENTITY default-password-cache                         "no">

<!-- DNS match not required by default in host-based authentication. -->
<!ENTITY default-auth-hostbased-require-dns-match       "no">

<!-- Keyboard-interactive authentication default failure delay in seconds. -->
<!ENTITY default-auth-kbdint-failure-delay              "2">

<!-- Keyboard-interactive authentication default maximum tries. -->
<!ENTITY default-auth-kbdint-max-tries                  "3">

<!-- Keyboard-interactive RADIUS server default port. -->
<!ENTITY default-radius-server-port                     "1812">
```

```
<!-- Keyboard-interactive RADIUS server default UDP recvfrom timeout. -->
<!ENTITY default-radius-server-timeout                  "10">

<!-- GSSAPI default ticket forwarding policy. -->
<!ENTITY default-gssapi-ticket-forwarding-policy        "no">

<!-- gssapi default library values. -->
<!ENTITY default-gssapi-dll-path "/usr/lib/libgssapi_krb5.so,/usr/lib64/libgssapi_krb5.so,/usr/lib/li

<!-- Default time in seconds for using expired CRLs. -->
<!ENTITY default-use-expired-crls                       "0">

<!-- CRLs are not disabled by default. -->
<!ENTITY default-disable-crls                           "no">

<!-- Digital signature in key usage is not enforced by default. -->
<!ENTITY default-dod-pki                                "no">

<!-- LDAP server default port. -->
<!ENTITY default-ldap-server-port                       "389">

<!-- Default CRL update minimum interval. -->
<!ENTITY default-crl-update-min-interval                "30">

<!-- Default interval for CRL prefetching. -->
<!ENTITY default-crl-prefetch-interval                  "3600">

<!-- Default crypto library mode ("fips" or "standard"). -->
<!ENTITY default-crypto-lib-mode                        "standard">

<!-- Both ipv4 and ipv6 are enabled by default -->
<!ENTITY default-address-family-type                    "inet">

<!-- Default terminate user started processes -->
<!ENTITY default-terminate-user-processes               "no">

<!ENTITY default-allow-configuration                    "no">

<!-- Default log event facility. -->
<!ENTITY default-log-event-facility                     "normal">

<!-- Default log event severity. -->
<!ENTITY default-log-event-severity                     "notice">

<!ENTITY default-access-action                          "allow">

<!-- Default value for the feature -->
<!ENTITY default-load-control-enable                    "yes">

<!-- Default value for the feature -->
<!ENTITY default-white-list-size                        "1000">

<!-- Default ignore AIX rlogin setting. -->
```

```
<!ENTITY default-ignore-aix-rlogin                      "no">

<!-- Default ignore AIX login setting. -->
<!ENTITY default-ignore-aix-login                       "no">

<!-- Default record sessions without PTYs. -->
<!ENTITY default-record-ptyless-sessions                "yes">

<!-- Default Windows logon type. -->
<!ENTITY default-windows-logon-type                     "interactive">

<!-- Default Windows terminal mode. -->
<!ENTITY default-windows-terminal-mode                  "console">

<!-- Default Ignore nisplus no permission error. -->
<!ENTITY default-ignore-nisplus-no-permission           "no">

<!-- TCP keepalives are disabled by default. -->
<!ENTITY default-tcp-keepalive                          "no">

<!-- Whether a plugin is allowed to not initialize (due to e.g. -->
<!-- system configuration, missing shared libraries).           -->
<!ENTITY default-allow-missing                          "no">

<!-- Default connection idle timeout in seconds.  The value zero -->
<!-- disables idle timeout. -->
<!ENTITY default-idle-timeout                           "0">

<!-- Message of the day (MOTD) is printed on login by default. -->
<!ENTITY default-print-motd                             "yes">

<!-- Authentication file permissions are checked by default. -->
<!ENTITY default-strict-modes                           "yes">

<!-- Default authentication file permission mask bits (octal). -->
<!ENTITY default-mask-bits                              "022">

<!-- Service name used with PAM. -->
<!ENTITY default-pam-service-name                       "ssh-server-g3">
<!-- Whether to perform PAM Account and Session management when executing -->
<!-- commands, i.e. shells, subsystems and remote commands.               -->
<!ENTITY default-pam-command-action                     "no">

<!-- Whether to bind x11 listeners to the localhost interface or to the   -->
<!-- 'any' interface. If the x11 listener is bound to the 'any' interface -->
<!-- the SO_REUSEADDR socket option will not be set.                      -->
<!ENTITY default-x11-listen-address                     "localhost">

<!-- Whether to only use PAM to check if the user is allowed to login.    -->
<!-- PAM can be used during authentication or via the                     -->
<!-- pam-calls-with-commands setting. If PAM is not used in either        -->
<!-- authentication or with pam-calls-with-commands the normal system     -->
<!-- checks will be used to determine whether the user is allowed to      -->
<!-- login i.e. account is not locked etc.                                -->
```

```
<!ENTITY default-pam-account-checking-only             "no">

<!-- Whether the server tries to resolve the client hostname during       -->
<!-- connection setup                                                     -->
<!ENTITY default-resolve-client-hostname               "yes">

<!-- Whether to suppress last login, password expiry, motd etc. messages  -->
<!-- during login.                                                        -->
<!ENTITY default-quiet-login                          "no">

<!-- Default certificate cache size in MBs. -->
<!ENTITY default-cert-cache-size                      "300">

<!-- Default CRL size limit (in MB). -->
<!ENTITY default-max-crl-size                         "50">

<!-- The default maximum path length for certificate validation. -->
<!ENTITY default-max-path-length                      "10">

<!-- Default timeout for external searches (LDAP, HTTP, OCSP) (seconds). -->
<!ENTITY default-external-search-timeout              "360">

<!-- Default limit of LDAP responses (MBs). -->
<!ENTITY default-max-ldap-response-length             "50">

<!-- Default LDAP connection idle timeout in seconds. -->
<!ENTITY default-ldap-idle-timeout                    "30">

<!-- Whether to enable AIX LAM password change by default. -->
<!ENTITY default-aix-lam-password-change              "no">

<!-- Keyboard-interactive RADIUS server default port. -->
<!ENTITY default-tunnel-mapper-timeout                "15">

<!-- Windows administrator is able to request elevated privileges. -->
<!ENTITY default-allow-elevation                      "yes">

<!-- Policy elements. -->

<!-- The top-level element. -->
<!ELEMENT secsh-server  (params?,connections?,authentication-methods?
                         ,services?)>

<!-- Parameter element. Only "hostkey" and "listener" are allowed multiple -->
<!-- times.                                                               -->
<!ELEMENT params (crypto-lib|address-family|hostkey|listener|settings|domain-policy
                 |logging|limits|cert-validation
                 |pluggable-authentication-modules|protocol-parameters|password-cache|
                 load-control|password-change-rules)*>

<!-- Cryptographic library. -->
<!ELEMENT crypto-lib    EMPTY>
<!ATTLIST crypto-lib
         mode          (fips|standard) "&default-crypto-lib-mode;">
```

```
<!-- address-family mode setting ipv4 & ipv6-->
<!ELEMENT address-family        EMPTY>
<!ATTLIST address-family
          type          (any|inet|inet6) "&default-address-family-type;">

<!-- Settings - a block for stuff that is too minor to have its
     own element in the params block. -->
<!ELEMENT settings      EMPTY>
<!ATTLIST settings
      signature-algorithms    CDATA    #IMPLIED
      proxy-scheme            CDATA    #IMPLIED
      xauth-path              CDATA    #IMPLIED
      xauth-shell             CDATA    #IMPLIED
      x11-listen-address      (localhost|any)
                                       "&default-x11-listen-address;"
      pam-account-checking-only (yes|no)
                                       "&default-pam-account-checking-only;"
      ignore-aix-rlogin       (yes|no) "&default-ignore-aix-rlogin;"
      ignore-aix-login        (yes|no) "&default-ignore-aix-login;"
      record-ptyless-sessions (yes|no) "&default-record-ptyless-sessions;"
      user-config-dir         CDATA    #IMPLIED
      default-path            CDATA    #IMPLIED
      windows-logon-type      (batch|interactive|network|network-cleartext)
                                       "&default-windows-logon-type;"
      windows-terminal-mode   (console|console-utf8|stream)
                                       "&default-windows-terminal-mode;"
      ignore-nisplus-no-permission (yes|no)
                                       "&default-ignore-nisplus-no-permission;"
      resolve-client-hostname (yes|no) "&default-resolve-client-hostname;"
      quiet-login             (yes|no) "&default-quiet-login;"
      default-domain          CDATA    #IMPLIED
      terminate-user-processes (yes|no) "&default-terminate-user-processes;"
      allow-elevation (yes|no) "&default-allow-elevation;">

<!ELEMENT pluggable-authentication-modules EMPTY>
<!ATTLIST pluggable-authentication-modules
          service-name           CDATA         "&default-pam-service-name;"
          dll-path               CDATA         #IMPLIED
          pam-calls-with-commands (yes|no|forced-no)
                                               "&default-pam-command-action;">

<!ELEMENT protocol-parameters EMPTY>
<!ATTLIST protocol-parameters
          threads CDATA #IMPLIED>

<!-- Hostkey specification. -->
<!ELEMENT hostkey       (((private,(public|x509-certificate|
                            openssh-certificate)?)
                  |externalkey)|x509-certificate-chain|
                        certificate-authority)*>
<!ATTLIST hostkey
   status        CDATA   #IMPLIED
   advertise     CDATA   #IMPLIED
```

```
   rotation-period CDATA #IMPLIED
   rotation-margin CDATA #IMPLIED>

<!-- Private key specification. -->
<!ELEMENT private       (#PCDATA)>
<!ATTLIST private
          file              CDATA   #IMPLIED
   passphrase        CDATA   #IMPLIED
   passphrase-file   CDATA   #IMPLIED
   passphrase-format CDATA   #IMPLIED>

<!-- Public key. -->
<!ELEMENT public        (#PCDATA)>
<!ATTLIST public
          file          CDATA   #IMPLIED>

<!-- Certificate (host). -->
<!ELEMENT x509-certificate      (#PCDATA)>
<!ATTLIST x509-certificate
          file          CDATA   #IMPLIED>

<!ELEMENT x509-certificate-chain (x509-certificate)*>

<!ELEMENT openssh-certificate      (#PCDATA)>
<!ATTLIST openssh-certificate
          file          CDATA   #IMPLIED>

<!-- External key. -->
<!ELEMENT externalkey   EMPTY>
<!ATTLIST externalkey
          type          CDATA   #REQUIRED
          init-info     CDATA   #IMPLIED>

<!ELEMENT certificate-authority (ca-external-command*)>
<!ATTLIST certificate-authority
          type                CDATA #IMPLIED
   name               CDATA #IMPLIED
   max-validity-period CDATA #IMPLIED
   log-file           CDATA #IMPLIED
   revocation-file    CDATA #IMPLIED>

<!ELEMENT ca-external-command   EMPTY>
<!ATTLIST ca-external-command
   type               CDATA #IMPLIED
   command            CDATA #IMPLIED
   args               CDATA #IMPLIED>

<!-- CA certificate. -->
<!ELEMENT ca-certificate        (#PCDATA)>
<!ATTLIST ca-certificate
          file                  CDATA           #IMPLIED
          name                  CDATA           #REQUIRED
          disable-crls          (yes|no)        "&default-disable-crls;"
          use-expired-crls      CDATA           "&default-use-expired-crls;"
```

```
          trusted               (yes|no)        "yes">

<!-- OpenSSH CA key. -->
<!ELEMENT openssh-ca-key        (#PCDATA)>
<!ATTLIST openssh-ca-key
          file                  CDATA           #IMPLIED
          name                  CDATA           #REQUIRED>

<!-- Certificate caching. -->
<!ELEMENT cert-cache-file       EMPTY>
<!ATTLIST cert-cache-file
          file                  CDATA   #REQUIRED>

<!-- CRL automatic updating. -->
<!ELEMENT crl-auto-update       EMPTY>
<!ATTLIST crl-auto-update
          update-before         CDATA   #IMPLIED
          minimum-interval      CDATA   "&default-crl-update-min-interval;">

<!-- CRL prefetch. -->
<!ELEMENT crl-prefetch          EMPTY>
<!ATTLIST crl-prefetch
          interval              CDATA   "&default-crl-prefetch-interval;"
          url                   CDATA   #REQUIRED>

<!-- LDAP server. -->
<!ELEMENT ldap-server           EMPTY>
<!ATTLIST ldap-server
          address               CDATA   #REQUIRED
          port                  CDATA   "&default-ldap-server-port;">

<!-- OCSP responder. -->
<!ELEMENT ocsp-responder        (#PCDATA)>
<!ATTLIST ocsp-responder
          validity-period       CDATA   #IMPLIED
          url                   CDATA   #REQUIRED
          certificate           CDATA   #IMPLIED>

<!-- Enforce digital signature in key usage. -->
<!ELEMENT dod-pki               EMPTY>
<!ATTLIST dod-pki
          enable                (yes|no)        "&default-dod-pki;">

<!-- Secure Shell server TCP listener address and port. -->
<!ELEMENT listener      EMPTY>
<!ATTLIST listener
          id            ID      #REQUIRED
          port          CDATA   "22"
          address       CDATA   #IMPLIED>


<!-- Server domain policy type -->
<!ELEMENT domain-policy                 (windows-domain)*>
<!ATTLIST domain-policy
```

```
          windows-domain-precedence     CDATA   #IMPLIED>

<!ELEMENT windows-domain      EMPTY>
<!ATTLIST windows-domain
    name    CDATA #REQUIRED
    user    CDATA #REQUIRED>

<!ELEMENT password-cache                        EMPTY>
<!ATTLIST password-cache
          file          CDATA   #REQUIRED>

<!-- Logging. -->
<!ELEMENT logging       (log-events*)>

<!-- Log events. -->
<!ELEMENT log-events    (#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;">

<!-- Certificate validation. Maximum one of each of "cert-cache-file", -->
<!-- "crl-auto-update" and "dod-pki" can be present.                    -->
<!ELEMENT cert-validation (ldap-server|ocsp-responder|cert-cache-file
                           |crl-auto-update|crl-prefetch|dod-pki
                           |ca-certificate|openssh-ca-key)*>

<!ATTLIST cert-validation
          http-proxy-url           CDATA   #IMPLIED
          socks-server-url         CDATA   #IMPLIED
          cache-size               CDATA   "&default-cert-cache-size;"
          max-crl-size             CDATA   "&default-max-crl-size;"
          external-search-timeout  CDATA   "&default-external-search-timeout;"
          max-ldap-response-length CDATA   "&default-max-ldap-response-length;"
          ldap-idle-timeout        CDATA   "&default-ldap-idle-timeout;"
          max-path-length          CDATA   "&default-max-path-length;">

<!ELEMENT access EMPTY>

<!ATTLIST access
          user                     CDATA   #REQUIRED
          action                   (allow|deny)      "&default-access-action;">


<!-- Limits. -->
<!-- max-connections is _per_servant_ .-->
<!-- servant-lifetime    - how many connections a servant will handle -->
<!-- before it is retired. -->

<!ELEMENT limits                  (servant-lifetime)*>
<!ATTLIST limits
```

```
          max-connections          CDATA   #IMPLIED
          max-processes            CDATA   #IMPLIED>

<!ELEMENT servant-lifetime         EMPTY>
<!ATTLIST servant-lifetime
          total-connections        CDATA   #IMPLIED>

<!ELEMENT load-control             EMPTY>
<!ATTLIST load-control
          enable                   (yes|no)      "&default-load-control-enable;"
          discard-limit            CDATA   #IMPLIED
          white-list-size          CDATA   "&default-white-list-size;">

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT password-change-rules  EMPTY>
<!ATTLIST password-change-rules
          allow-configuration   (yes|no) "&default-allow-configuration;">

<!-- Connections. -->
<!ELEMENT connections   (connection+)>

<!-- Connection. -->
<!ELEMENT connection    (selector*,rekey?,cipher*,mac*,kex*,hostkey-algorithm*,compression*)>
<!ATTLIST connection
          name          ID                      #IMPLIED
          action        (allow|deny)            "&default-connection-action;"
          tcp-keepalive (yes|no)                "&default-tcp-keepalive;">

<!-- Rekey intervals. -->
<!ELEMENT rekey         EMPTY>
<!ATTLIST rekey
          seconds       CDATA   "&default-rekey-interval-seconds;"
          bytes         CDATA   "&default-rekey-interval-bytes;">

<!-- Cipher. -->
<!ELEMENT cipher        EMPTY>
<!ATTLIST cipher
          name          CDATA                   #REQUIRED
          allow-missing (yes|no)                "&default-allow-missing;">

<!-- MAC. -->
<!ELEMENT mac           EMPTY>
<!ATTLIST mac
          name          CDATA                   #REQUIRED
          allow-missing (yes|no)                "&default-allow-missing;">

<!-- KEX. -->
<!ELEMENT kex           EMPTY>
<!ATTLIST kex
          name          CDATA                   #REQUIRED
          allow-missing (yes|no)                "&default-allow-missing;">

<!-- Hostkey algorithm. -->
<!ELEMENT hostkey-algorithm  EMPTY>
```

```
<!ATTLIST hostkey-algorithm
          name          CDATA                   #REQUIRED
          allow-missing (yes|no)                "&default-allow-missing;">

<!-- Compression. -->
<!ELEMENT compression  EMPTY>
<!ATTLIST compression
          name         CDATA                    #IMPLIED>

<!-- Selector element. -->
<!ELEMENT selector      (interface|certificate|host-certificate|ip
                         |user|user-group|user-privileged|blackboard
                         |publickey-passed|user-password-change-needed)*>

<!-- Interface selector. At least one parameter must be given. If id is -->
<!-- set, the others MUST NOT be set. If id is not set, either or both  -->
<!-- of address and port may be defined.                                -->
<!ELEMENT interface       EMPTY>
<!ATTLIST interface
          id              IDREF    #IMPLIED
          address         CDATA    #IMPLIED
          port            CDATA    #IMPLIED
          allow-undefined (yes|no) "&default-allow-undefined-value;">

<!-- Public key (plain) passed selector. -->
<!ELEMENT publickey-passed      EMPTY>
<!ATTLIST publickey-passed
          type                  CDATA    #IMPLIED
          length                CDATA    #IMPLIED
          allow-undefined       (yes|no)
                                "&default-allow-undefined-value;">

<!-- Certificate selector. -->
<!ELEMENT certificate   EMPTY>
<!ATTLIST certificate
          field           (ca-list|issuer-name|subject-name|serial-number
                           |altname-email|altname-upn
                           |altname-ip|altname-fqdn
                           |extended-key-usage
                           |openssh-principals|openssh-cert-type
                           |openssh-extension|openssh-key-id) #REQUIRED
          pattern                CDATA  #IMPLIED
          pattern-case-sensitive CDATA  #IMPLIED
          regexp                 CDATA  #IMPLIED
          ignore-prefix          (yes|no) #IMPLIED
          ignore-suffix          (yes|no) #IMPLIED
          explicit               (yes|no) #IMPLIED
          allow-undefined        (yes|no)
                                 "&default-allow-undefined-value;">

<!-- Host certificate selector. -->
<!ELEMENT host-certificate      EMPTY>
<!ATTLIST host-certificate
          field           (ca-list|issuer-name|subject-name|serial-number
```

```
                        |altname-email|altname-upn
                        |altname-ip|altname-fqdn
                        |extended-key-usage
                    |openssh-principals|openssh-cert-type
                        |openssh-extension|openssh-key-id) #REQUIRED
          pattern               CDATA  #IMPLIED
          pattern-case-sensitive CDATA  #IMPLIED
          regexp                CDATA  #IMPLIED
          ignore-prefix         (yes|no) #IMPLIED
          ignore-suffix         (yes|no) #IMPLIED
          explicit              (yes|no) #IMPLIED
          allow-undefined       (yes|no)
                                "&default-allow-undefined-value;">

<!-- IP address selector. -->
<!-- The address will be one of the following:                          -->
<!--   - an IP range of the form x.x.x.x-y.y.y.y                        -->
<!--   - an IP mask of the form x.x.x.x/y                               -->
<!--   - a straight IP address x.x.x.x                                  -->
<!--   - an FQDN pattern (form not checked, either it matches or not)   -->
<!-- Exactly one of address or fqdn must be set. -->
<!ELEMENT ip            EMPTY>
<!ATTLIST ip
          address               CDATA   #IMPLIED
          fqdn                  CDATA   #IMPLIED
          fqdn-regexp           CDATA   #IMPLIED
          allow-undefined       (yes|no)
                                "&default-allow-undefined-value;">

<!-- User name selector. -->
<!ELEMENT user                  EMPTY>
<!ATTLIST user
          name                  CDATA   #IMPLIED
          name-case-sensitive   CDATA   #IMPLIED
          name-regexp           CDATA   #IMPLIED
          id                    CDATA   #IMPLIED
          allow-undefined       (yes|no)
                                "&default-allow-undefined-value;">

<!-- User group selector. -->
<!ELEMENT user-group            EMPTY>
<!ATTLIST user-group
          name                  CDATA   #IMPLIED
          name-case-sensitive   CDATA   #IMPLIED
          name-regexp           CDATA   #IMPLIED
          id                    CDATA   #IMPLIED
          allow-undefined       (yes|no)
                                "&default-allow-undefined-value;">

<!-- User privileged (administrator) selector. -->
<!ELEMENT user-privileged       EMPTY>
<!ATTLIST user-privileged
          value                 (yes|no)
                                "&default-user-privileged-value;"
```

```
          allow-undefined       (yes|no)
                                "&default-allow-undefined-value;">

<!-- Selector for the need of user password change. -->
<!ELEMENT user-password-change-needed   EMPTY>
<!ATTLIST user-password-change-needed
          value                 (yes|no)
                                "&default-user-password-change-needed-value;"
          allow-undefined       (yes|no)
                                "&default-allow-undefined-value;">

<!-- Blackboard selector. -->
<!ELEMENT blackboard            EMPTY>
<!ATTLIST blackboard
          field                         CDATA   #REQUIRED
          pattern                       CDATA   #IMPLIED
          pattern-case-sensitive        CDATA   #IMPLIED
          regexp                        CDATA   #IMPLIED
          allow-undefined               (yes|no)
                                        "&default-allow-undefined-value;">


<!-- Authentication methods element. -->
<!ELEMENT authentication-methods        (banner-message?,auth-file-modes?
                                         ,authentication*)>
<!ATTLIST authentication-methods
          login-grace-time      CDATA   "&default-login-grace-time-seconds;">

<!-- Banner message element. -->
<!ELEMENT banner-message        (#PCDATA)>
<!ATTLIST banner-message
          file          CDATA   #IMPLIED>

<!-- Authentication file permission checks. -->
<!ELEMENT auth-file-modes       EMPTY>
<!ATTLIST auth-file-modes
          strict                (yes|no)        "&default-strict-modes;"
          mask-bits             CDATA           "&default-mask-bits;"
          dir-mask-bits         CDATA           #IMPLIED>

<!-- Authentication element.  In an authentication element, different -->
<!-- authentication methods are in OR-relation.  User must pass one of -->
<!-- them. -->
<!ELEMENT authentication        (selector*
                                 ,(set-blackboard|login-restrictions)*
                                 ,(auth-publickey|auth-hostbased|auth-password
                                   |auth-keyboard-interactive|auth-gssapi)*
                                 ,mapper?
                                 ,set-user?
                                 ,authentication*)>
<!ATTLIST authentication
        name            ID              #IMPLIED
          action        (allow|deny)    "&default-authentication-action;"
          set-group     CDATA           #IMPLIED
```

```
          repeat-block  (yes|no)        "no"
          password-cache (yes|no)       "&default-password-cache;" >



<!ELEMENT set-user      EMPTY>
<!ATTLIST set-user
          name          CDATA           #REQUIRED>

<!ELEMENT mapper        EMPTY>
<!ATTLIST mapper
          command       CDATA           #REQUIRED
          timeout       CDATA           "&default-tunnel-mapper-timeout;"
          chroot        CDATA           #IMPLIED
          user          CDATA           #IMPLIED >

<!ELEMENT login-restrictions EMPTY>
<!ATTLIST login-restrictions
          ignore-password-expiration    CDATA #IMPLIED
          ignore-aix-rlogin             CDATA #IMPLIED
          ignore-aix-login              CDATA #IMPLIED
          ignore-nisplus-no-permission  CDATA #IMPLIED>

<!ELEMENT set-blackboard                (#PCDATA)>
<!ATTLIST set-blackboard
          field                         CDATA #REQUIRED
          value                         CDATA #IMPLIED
          file                          CDATA #IMPLIED>

<!-- Public-key authentication. -->
<!ELEMENT auth-publickey        EMPTY>
<!ATTLIST auth-publickey
          require-dns-match             (yes|no)
                                        "&default-auth-publickey-require-dns-match;"
          signature-algorithms          CDATA #IMPLIED
          authorization-file            CDATA #IMPLIED
          authorized-keys-directory     CDATA #IMPLIED
          openssh-authorized-keys-file  CDATA #IMPLIED
          allow-missing                  (yes|no)
                                         "&default-allow-missing;">

<!-- Host-based authentication. -->
<!ELEMENT auth-hostbased        EMPTY>
<!ATTLIST auth-hostbased
          require-dns-match     (yes|no)
                                "&default-auth-hostbased-require-dns-match;"
          disable-authorization (yes|no) "no"
          allow-missing         (yes|no)
                                "&default-allow-missing;">

<!-- Password authentication. -->
<!ELEMENT auth-password         EMPTY>
<!ATTLIST auth-password
          failure-delay         CDATA "&default-auth-password-failure-delay;"
```

```
          max-tries             CDATA "&default-auth-password-max-tries;"
          use-pam               (yes|no) #IMPLIED
          allow-missing         (yes|no) "&default-allow-missing;" >

<!-- Keyboard-interactive authentication. -->
<!ELEMENT auth-keyboard-interactive     ((submethod-pam
                                          |submethod-password
                                          |submethod-securid
                                          |submethod-radius
                                          |submethod-aix-lam
                                          |submethod-generic)*)>

<!ATTLIST auth-keyboard-interactive
          failure-delay         CDATA "&default-auth-kbdint-failure-delay;"
          max-tries             CDATA "&default-auth-kbdint-max-tries;"
          allow-missing         (yes|no) "&default-allow-missing;" >

<!-- Keyboard-interactive submethods. -->

<!-- PAM. service-name is #IMPLIED, as it will be by default whatever is -->
<!-- set in "params" block.                                              -->
<!ELEMENT submethod-pam         EMPTY>
<!ATTLIST submethod-pam
          service-name          CDATA   #IMPLIED
          dll-path              CDATA   #IMPLIED
          allow-missing         (yes|no) #IMPLIED>

<!-- Password. -->
<!ELEMENT submethod-password    EMPTY>
<!ATTLIST submethod-password
          allow-missing         (yes|no) #IMPLIED>

<!-- SecurID. -->
<!ELEMENT submethod-securid     EMPTY>
<!ATTLIST submethod-securid
          dll-path              CDATA   #IMPLIED
          allow-missing         (yes|no) #IMPLIED>

<!-- RADIUS. -->
<!ELEMENT submethod-radius      (radius-server+)>
<!ATTLIST submethod-radius
          allow-missing         (yes|no) #IMPLIED>

<!-- RADIUS server. -->
<!ELEMENT radius-server         (radius-shared-secret)>
<!ATTLIST radius-server
          address               CDATA   #REQUIRED
          port                  CDATA   "&default-radius-server-port;"
          timeout               CDATA   "&default-radius-server-timeout;"
          client-nas-identifier CDATA   #IMPLIED>

<!-- Secret. "file" has precedence over #PCDATA. -->
<!ELEMENT radius-shared-secret  (#PCDATA)>
<!ATTLIST radius-shared-secret
```

```
          file                  CDATA   #IMPLIED>

<!-- AIX LAM. -->
<!ELEMENT submethod-aix-lam      EMPTY>
<!ATTLIST submethod-aix-lam
          enable-password-change (yes|no) "&default-aix-lam-password-change;"
          allow-missing         (yes|no) #IMPLIED>

<!-- Generic submethod. -->
<!ELEMENT submethod-generic     EMPTY>
<!ATTLIST submethod-generic
          name                  CDATA   #REQUIRED
          params                CDATA   #IMPLIED
          allow-missing         (yes|no) #IMPLIED>

<!-- GSSAPI authentication. -->
<!ELEMENT auth-gssapi   EMPTY>
<!ATTLIST auth-gssapi
          dll-path                    CDATA     "&default-gssapi-dll-path;"
          allow-ticket-forwarding     (yes|no)
                                      "&default-gssapi-ticket-forwarding-policy;"
          allow-missing               (yes|no)
                                      "&default-allow-missing;">

<!-- Services element. -->
<!ELEMENT services      (group*,rule+)>

<!-- Group element. -->
<!ELEMENT group         (selector+)>
<!ATTLIST group
          name          ID      #REQUIRED>

<!-- Rule element. Maximum one of each of "terminal", "tunnel-agent"    -->
<!-- or "tunnel-x11" can be present.                                    -->
<!ELEMENT rule          (environment|terminal|subsystem|command
                         |tunnel-agent|tunnel-x11|tunnel-local
                         |tunnel-remote)*>

<!-- "group", if defined, will be used to match the rule. -->
<!ATTLIST rule
          group         CDATA           #IMPLIED
          idle-timeout  CDATA           "&default-idle-timeout;"
          print-motd    (yes|no)        "&default-print-motd;">

<!-- Environment. -->
<!-- The default allowed environment variables are:            -->
<!-- allowed-case-sensitive="TERM,PATH,TZ,LANG,LC_*"           -->
<!-- If neither allowed nor allowed-case-sensitive is set,     -->
<!-- the default is used.                                      -->
<!ELEMENT environment   EMPTY>
<!ATTLIST environment
          allowed                       CDATA   #IMPLIED
          allowed-case-sensitive        CDATA   #IMPLIED>
```

```
<!-- Terminal. -->
<!ELEMENT terminal      EMPTY>
<!ATTLIST terminal
          action        (allow|deny)            "&default-terminal-action;"
          chroot        CDATA                   #IMPLIED>

<!-- Subsystem. -->
<!ELEMENT subsystem     (attribute*)>
<!ATTLIST subsystem
          type          CDATA           #REQUIRED
          action        (allow|deny)    "&default-subsystem-action;"
          audit         (yes|no)        "&default-subsystem-audit;"
          exec-directly CDATA   #IMPLIED
          application   CDATA           #IMPLIED
          chroot        CDATA           #IMPLIED
          pass-bb       (yes|no)        "no">

<!ELEMENT attribute     EMPTY>
<!ATTLIST attribute
          name          CDATA   #REQUIRED
          value         CDATA   #IMPLIED>

<!-- Tunnels. -->

<!ELEMENT tunnel-x11    EMPTY>
<!ATTLIST tunnel-x11
          action        (allow|deny)            "&default-tunnel-action;">

<!ELEMENT tunnel-agent  EMPTY>
<!ATTLIST tunnel-agent
          action        (allow|deny)            "&default-tunnel-action;">

<!ELEMENT tunnel-local  (mapper|((src|tunnel-src|dst)*))>
<!ATTLIST tunnel-local
          action        (allow|deny)            "&default-tunnel-action;">

<!ELEMENT tunnel-remote ((src|tunnel-dst|listen)*)>
<!ATTLIST tunnel-remote
          action        (allow|deny)            "&default-tunnel-action;"
          disable-privilege-check (yes|no) "no">

<!-- Tunnel selectors. These apply only to TCP local and remote tunnels.-->
<!-- src and dst are for local-tcp                                      -->
<!-- src and listen are for remote-tcp                                  -->

<!-- address or fqdn are not mandatory. If set, exactly one must be set -->
<!-- (not both).                                                        -->

<!-- Source. -->

<!ELEMENT src           EMPTY>
<!ATTLIST src
          address       CDATA   #IMPLIED
          fqdn          CDATA   #IMPLIED
```

```
          fqdn-regexp   CDATA   #IMPLIED
          port          CDATA   #IMPLIED>

<!-- Destination. -->
<!ELEMENT dst           EMPTY>
<!ATTLIST dst
          address       CDATA   #IMPLIED
          fqdn          CDATA   #IMPLIED
          fqdn-regexp   CDATA   #IMPLIED
          port          CDATA   #IMPLIED>

<!-- Listener. -->
<!ELEMENT listen        EMPTY>
<!ATTLIST listen
          address       CDATA   #IMPLIED
          port          CDATA   #IMPLIED>

<!-- Tunnel source. -->
<!ELEMENT tunnel-src           EMPTY>
<!ATTLIST tunnel-src
          address       CDATA   #IMPLIED
          fqdn          CDATA   #IMPLIED
          fqdn-regexp   CDATA   #IMPLIED>

<!ELEMENT tunnel-dst           EMPTY>
<!ATTLIST tunnel-dst
          address       CDATA   #IMPLIED
          fqdn          CDATA   #IMPLIED
          fqdn-regexp   CDATA   #IMPLIED>

<!-- Command. -->
<!ELEMENT command                     EMPTY>
<!ATTLIST command
          action                      (allow|deny|forced)
                                             "&default-command-action;"
          interactive                 (yes|no)
                                             "&default-interactive-command-action;"
          application                 CDATA   #IMPLIED
          application-case-sensitive  CDATA   #IMPLIED
          chroot                      CDATA   #IMPLIED>
```

Appendix C Command-Line Tools and Man Pages

Tectia Server 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-server-g3(8): Tectia Server – Generation 3
- ssh-server-ctl(8): Tectia Server control utility
- ssh-troubleshoot(8): utility for collecting system information for troubleshooting purposes
- 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 a description of the Tectia Server configuration file options, see ssh-server-config(5), the configuration file format.

ssh-server-g3

ssh-server-g3 — Secure Shell server - Generation 3

Synopsis

`ssh-server-g3` [`-4`] [`-6`] [`-D, --debug=` *`LEVEL`*] [`-f, --config-file=` *`FILE`*] [`-H, --hostkey=` *`FILE`*]
[`-l, --listen=` [*`ADDRESS:`*] *`PORT`*] [`-n, --num-processes=` *`NUM`*]
[`--auxdata-path=` *`PATH`*] [`--fips-mode` [`=yes` | `no`]]
[`--libexec-path=` *`PATH`*] [`--max-num-processes=` *`NUM`*] [`--plugin-path=` *`PATH`*]
[`-V, --version`] [`-h, --help`]

Description

ssh-server-g3 is the Secure Shell server program for Tectia Server.

The **ssh-server-g3** command should not be used directly, except for debugging purposes. Use instead the startup script with the same name, **ssh-server-g3**.

The path to the **ssh-server-g3** startup script varies between operating systems:

- On Linux with systemd:

```
# systemctl [command] ssh-server-g3
```

- On Solaris, and Linux without systemd:

```
# /etc/init.d/ssh-server-g3 [command]
```

- On HP-UX:

```
# /sbin/init.d/ssh-server-g3 [command]
```

Supported commands:

start

Start the server.

stop

Stop the server. Existing connections stay open until closed from the client side.

restart

Start a new server process. Existing connections stay open using the old server process. The old process is closed after the last old connection is closed from the client side.

reload

Reload the configuration file. Existing connections stay open.

On AIX platforms, use the System Resource Controller (SRC) of the operating system to stop and start the server process manually.

To start Tectia Server on AIX, enter command:

```
startsrc -s ssh-tectia-server
```

To stop Tectia Server on AIX, enter command:

```
stopsrc -s ssh-tectia-server
```

On AIX, using **startsrc** starts two ssh-server-g3 processes. One process is so-called service launcher that interfaces with the SRC and the actual SSH server process. By using a separate service launcher, the SRC is able to start a new server process in the case that old server process has been stopped but it is still serving open connections.

On Windows, use the Tectia Server Configuration GUI or the Windows Services console to stop and start the server process.

Options

When the **ssh-server-g3** command is used directly, it accepts the following options:

`-4`

Accepts only IPv4 connections and works in IPv4 mode.

`-6`

Accepts only IPv6 connections and works in IPv6 mode.

`-D, --debug=` *`LEVEL`*

Sets the debug level string to *`LEVEL`*.

`--direct`

Required on Linux with systemd for enabling the following options: `--listen`, `--port`, `-p`, `-l`, `--old`, `--pid`, `--server-address`. Note that systemd cannot correctly track the server status when using the `--direct` option.

`-f, --config-file=` *`FILE`*

Reads the Tectia Server configuration file from *`FILE`* instead of the default location.

`-H, --hostkey=` *`FILE`*

Specifies the host key file to be used.

`-l, --listen=` [*`ADDRESS:`*]*`PORT`*

Specifies the listen address and port. If *`ADDRESS`* is unspecified, listen on any IP address. If IPv6 address is used, the address must be inside brackets, for example, `[::1]`.

`-n, --num-processes=` *`NUM`*

Sets the number of Servant processes to *`NUM`*. This value defines also the maximum number of Servants that the master server is allowed to have running at a time.

`--auxdata-path=` *`PATH`*

Sets the path to the auxiliary data directory.

`--fips-mode` [`=yes` | `no`]

When set to `yes`, uses the FIPS mode for the cryptographic library. When set to `no`, uses the standard mode for the cryptographic library. If the option is given without the `yes|no` argument, `yes` is assumed. If the option is not given at all on the command line, the mode specified in the `ssh-server-config.xml` file is used (by default, the standard mode).

`--libexec-path=` *`PATH`*

Sets the path to the `libexec` directory.

`--max-num-processes=` *`NUM`*

Sets the maximum number of Servant processes to *`NUM`*.

`--plugin-path=` *`PATH`*

Sets the path to the plugin directory.

`-V, --version`

Displays program version and exits.

`-h, --help`

Displays a short summary of command-line options and exits.

Login Process

When a user logs in successfully, **ssh-server-g3** does the following:

1. Changes process to run with normal user privileges.
2. Sets up the basic environment.
3. *(On Solaris)* Reads `/etc/default/login`, if it exists.
4. *(On Unix)* Reads `/etc/environment`, if it exists.
5. *(On Unix)* Reads `$HOME/.ssh2/environment`, if it exists.

   Note that setting the environment variables included in this file on the client side must be allowed in the Tectia Server configuration using the **`environment`** element.

6. Changes to the user's home directory.

7. Checks for RC files and runs it from the user's home directory (by default `$HOME/.ssh2/rc`) or, if that does not exist, runs `/etc/ssh2/sshrc`. Any RC file stored in the user's home directory will be run with the user's shell, and any global RC file will be run with `/bin/sh`.

8. Runs the user's shell, or the specified command or subsystem.

Environment Variables

Upon connection, Tectia Server will automatically set a number of environment variables that can be used by Secure Shell clients. The clients can also set or change the value of the environment variables if allowed by the server configuration (`ssh-server-config.xml`). The following variables are set by **ssh-server-g3**:

`DISPLAY` *(Unix)*

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 or equal than 1. Secure Shell clients use this special value to forward X11 connections over the secure channel.

`HOME` *(Unix)*

The user's home directory.

`LOGNAME` *(Unix)*

Synonym for `USER`; set for compatibility with systems using this variable.

`MAIL` *(Unix)*

The user's mailbox.

`PATH` *(Unix)*

Set to the default PATH, depending on the operating system or, on some systems, `/etc/environment` or `/etc/default/login`.

`SSH_SOCKS_SERVER` *(Unix)*

The address of the SOCKS server used by the client.

`SSH2_AUTH_SOCK` *(Unix)*

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` *(Unix)*

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, SSH_ORIGINAL_COMMAND`

This will be the original command given to the Secure Shell client 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` *(Unix)*

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.

`TERM`

The terminal type of the Secure Shell client.

`TZ` *(Unix)*

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` *(Unix)*

The name of the user.

Files

ssh-server-g3 uses the following files:

Note

`<INSTALLDIR>` indicates the default Tectia installation directory on Windows:

- `"C:\Program Files (x86)\SSH Communications Security\SSH Tectia"` on 64-bit Windows versions

`/etc/ssh2/ssh-server-config.xml`

This is the **ssh-server-g3** configuration file. The format of this file is described in ssh-server-config(5).

On Windows, the configuration file is located in `"<INSTALLDIR>\SSH Tectia Server\ssh-server-config.xml"`.

`/etc/ssh2/hostkey[.pub|.pass]`

These files are the default host key pair used by Tectia Server for authenticating itself to the clients. A 3072-bit RSA key pair is automatically generated during a fresh installation. It consists of the private key (`hostkey`) and the public key (`hostkey.pub`), and a passphrase file (`hostkey.pass`) if the private key has been encrypted with a random passphrase.

On Windows, the default host key pair and random passphrase is located in "<INSTALLDIR>\SSH Tectia Server\hostkey[.pub|.pass]".

/etc/ssh2/random_seed

This file is used for seeding the random number generator. 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 RHEL, the random seed file is located in "/var/opt/tectia/random_seed".

On Windows, the random seed file is located in "<INSTALLDIR>\SSH Tectia Server \random_seed".

/etc/ssh2/trusted_hosts

This directory is for storing the client host public keys that are trusted for host-based authentication.

The public-key files should be named according to the following pattern:

```
<hostname>.<keytype>.pub
```

In the key name, *<hostname>* is the hostname the client is sending to the server and *<keytype>* is the type of the public key (ssh-dss, ssh-rsa, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521). For example, a key called *client.example.com.ssh-dss.pub* is a DSS key that is trusted for login from the host *client.example.com*.

On Windows, the trusted host key directory is located in "<INSTALLDIR>\SSH Tectia Server \trusted_hosts".

For more information, see Section 5.8.

$HOME/.ssh2/authorized_keys (user-specific)

This directory is the default location used for the user public keys that are authorized for login.

On Windows, the default directory is %D/.ssh2/authorized_keys where %D expands to user's home directory, typically C:\Users\<username>.

$HOME/.ssh2/authorization (user-specific)

This is the default file that lists the user public keys that are authorized for login.

Using the authorization file is optional. If the file does not exist, Tectia Server looks for authorized keys in the $HOME/.ssh2/authorized_keys directory, by default, or in another authorized-keys directory defined in the Tectia Server configuration.

The authorization file contains a list of public key filenames each preceded by the keyword Key, and each one on its own line. All public keys listed in the authorization file are authorized for login. An example file is shown below:

```
Key         mykey.pub
```

This directs Tectia Server to use `$HOME/.ssh2/mykey.pub` as a valid public key when authorizing login.

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. The directory path can also contain a pattern string that is expanded by Tectia Server.

The following pattern strings can be used:

- `%D` is the user's home directory
- `%U` is the user's login name; expands to `domain.user` with Windows domain users.
- `%IU` is the user's user ID (uid); not supported on Windows
- `%IG` is the user's group ID (gid); not supported on Windows

Examples of allowed key paths are shown below:

```
Key authorized_keys/key1.pub
Key /tmp/key2.pub
Key /usr/%U/key3.pub
```

Optionally, additional parameters can be specified for the keys by using the `Options` keyword. See the section called “Authorization File Options” for more information.

On Windows, the default authorization file is located in user's home directory `%D/.ssh2/authorization`. Key paths in the file can be absolute or relative to the `C:\Users\<username>\.ssh2` directory.

`$HOME/.ssh/authorized_keys` (user-specific)

This is the default file used by OpenSSH server that contains the user public keys that are authorized for login. It is supported also by Tectia Server from version 5.1 onwards. The location of the file must be defined in the `ssh-server-config.xml` file by using the `openssh-authorized-keys-file` attribute. See **auth-publickey** .

The file contains public keys, one on each row, and options. The format of each row is as follows:

```
options  keytype  base64-encoded-key  comment
```

Tectia Server supports all OpenSSH-style `authorized_keys` file options, except `permitopen="host:port"` and `tunnel="n"`.

For more information on the format of this file, see the OpenSSH sshd(8) man page.

Authorization File Options

On the first line of the authorization file, you can optionally specify the regular expression syntax that is used when parsing hostname patterns in the `allow-from` and `deny-from` options (see below). The format of the first line is as follows:

```
## REGEX-SYNTAX egrep
```

The value for the syntax can be `egrep` (default), `ssh`, `zsh_fileglob`, or `traditional`. The values are not case-sensitive. `zsh_fileglob` and `traditional` are synonymous.

Note

The Tectia Server implementation of matching and parsing the patterns does not fully behave as `egrep`, because in the case of `egrep` and `ssh` REGEX syntax, Tectia Server encloses the pattern with `^(<existing_pattern>)$`, and therefore the parsed string must have the same length as the text to match.

For each key in the `authorization` file, options can be specified using the `Options` keyword. This keyword, if used, must follow the `Key` keyword above. Multiple options must be specified as a comma-separated list on one line. Tectia Server supports the following options:

`allow-from` and `deny-from`

In addition to public-key authentication, the canonical name of the remote host must match the given pattern(s). You can use a host name or an IP address to specify the remote host. Enter the host name or IP address with or without quotation marks.

Note

Because of Tectia Server's implementation for handling `egrep` REGEX syntax, when providing host names and IP addresses as pattern(s), use the escape character for periods, for instance host.example.com should be given as pattern `host\.example\.com`.

Specify one pattern per keyword; multiple keywords can be used. See the example below.

If you specify host names in the `allow-from` or `deny-from` options, ensure that you set the following attribute values in the server configuration file `ssh-server-config.xml`:

- `require-dns-match="yes"` under the `auth-publickey` element
- `resolve-client-hostname="yes"` under the `settings` element

These settings will prevent authentications from failing in case of problems with DNS lookups and reverse mapping.

Note

The authorization file is read first to find the keys allowed for authentication. The `deny-from` option will not work if the key is stored in the `authorized_files` directory or other location later in the reading order defined by the `auth-publickey` element (see **auth-publickey**) in `ssh-server-config.xml`.

`command="command"`

This is used to specify a "forced command" that will be executed on the server side instead of anything else when the user is authenticated. The command supplied by the user (if any) is put in

the environment variable `SSH2_ORIGINAL_COMMAND`. The command is run on a pty if the connection requests a pty; otherwise it is run without a tty. Quotes may be used in the command if escaped with backslashes.

This option is useful for restricting certain public keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Notice that the client may specify TCP/IP and/or X11 forwarding, unless they are explicitly denied (see `no-port-forwarding` and `no-x11-forwarding` below).

If terminal is explicitly allowed in the `ssh-server-config.xml` file, the forced command is run only when the user tries to run remote commands. If the user requests a shell, he can get it normally and the forced command is not run.

If a forced command is defined in the `ssh-server-config.xml` file, it overrides any commands in the authorization files. The configuration file might also allow only specific commands, or deny all remote commands. These restrictions apply also to commands in the authorization file.

For more information on command restrictions in the configuration file, see **command** .

`environment="NAME=value"`

This option specifies that the string is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted.

`idle-timeout="time"`

This option sets idle timeout limit to time either in seconds (`s` or nothing after the number), in minutes (`m`), in hours (`h`), in days (`d`), or in weeks (`w`). If the connection has been idle (all channels) this long, the connection is closed.

`no-port-forwarding`

This option forbids TCP/IP forwarding when this key is used for authentication. Any port forward (tunneling) requests by the client will return an error. This is useful in combination with the `command` option.

`no-x11-forwarding`

This option forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

`no-agent-forwarding`

This option forbids authentication agent forwarding when this key is used for authentication.

`no-pty`

This option prevents tty allocation (a request to allocate a pty will fail).

An example of an authorization file is shown below:

```
## REGEX-SYNTAX egrep

# First key: login allowed only from the specified IP address
Key     key1.pub
Options allow-from="10\.1\.100\.1", command="echo FOOBAR", no-x11-forwarding

# Second key: login allowed and denied only from the specified IP addresses
Key     key2.pub
Options allow-from="10\.1\.100\.2", deny-from="10\.1\.100\.1"

# Third key: forced command for doing a backup of the disk drive
Key     key3.pub
Options command="dd if=/dev/hda", no-port-forwarding, no-x11-forwarding
```

ssh-server-ctl

ssh-server-ctl — Tectia Server control utility.

Synopsis

`ssh-server-ctl [ options ] [ command [ command-options ] ]`

Description

ssh-server-ctl (**ssh-server-ctl.exe** on Windows) is a control utility that can be used to start, stop, or reload the configuration of Tectia Server (**ssh-server-g3**). It can also be used to add new servants or to stop servants, to check the status of the server, enable or disable debug mode on the running ssh-server-g3, servants and/or user SFTP server processes or to pause the server. Furthermore, on Windows it can be used for password cache management.

Note

ssh-server-ctl (**ssh-server-ctl.exe** on Windows) must be run as a privileged user (root or administrator).

To use the server control utility on Windows command-line, the *Windows PowerShell* or `cmd.exe` has to be started with *Run as Administrator* and the control utility executed from its install directory "`<INSTALLDIR>\SSH Tectia Server\`", for example:

```
C:\CustomDir\SSH Tectia Server\Tectia> .\ssh-server-ctl.exe status
```

Options

The following options are available:

`-C, --current`

Connects to the current server.

`-D, --debug=` *`LEVEL`*

Defines the debug level for ssh-server-ctl utility itself.

`-h, --help`

Displays the help text for the command.

`-l, --listen=` *`PORT`*

Same as the `port` option.

`-O, --old`

Connects to the previous retired server.

`-P, --pid PID`

Targets the command to the **ssh-server-g3** process identified with the given *PID*. Available on Unix only.

`-p, --port PORT`

Targets the command to the **ssh-server-g3** process running on the given *PORT*. The default port 22 is assumed if this option is not used.

`--server-address`

The path to the server control socket.

`-q, --quiet`

Displays little or no output depending on the command.

`-s, --short`

Displays a shorter more machine readable output.

`-v, --verbose`

Displays more information if it is available.

`-V, --version`

Displays the version string.

Commands

ssh-server-ctl (**ssh-server-ctl.exe** on Windows) accepts the following commands:

`add-servant`

Start a new servant or new servants.

Options:

`num`

Defines the number of servants to be started.

`continue`

Continue a previously paused service.

`debug`

Debug the running service. Sets or clears debug level in running server or reads debug information from it.

Usage: **ssh-server-ctl debug [options] <command> [<debug_level>]**

The 'set' and 'clear' commands are directed to the main server and all servants. New servants will get the same debug level the main server has. By using **'--servant <id|'active'|'current'>'** option before command the debug level will be set only to some servant. The 'active' means all running, not retired, servants. The 'current' is the servant that previously accepted new connection. The 'id' is the servant ID number from **ssh-server-ctl status** output.

Note

The levels 1-9 are the recommended debug levels. If level over 4 is needed, it is recommended to set only the specific module(s) with higher debug level, for example `"4,SshUser*=9"`

If debug is enabled, remember to disable it with **ssh-server-ctl debug clear** command. It should always be used after reproducing the problem. If the service has been restarted since debug mode was enabled on Unix, the **ssh-server-ctl --old debug clear** is needed to clear the debug settings from any retired server processes.

Commands:

`set <debug_level>`

The 'set' command sets the debug level to running server and all servants connected to it.

`set-one <debug_level>`

The 'set-one' command starts a new servant and sets the debug level only to it. The debug is only enabled for the servant that is going to process the next incoming connection. Note that this will increase the preferred number of servants.

`clear`

The 'clear' will reset the debug level and also close all debug hooks. The level only can be cleared by setting it to '0'. Use with 'ssh-server-ctl' option `--old` to clear debug settings from previous, retired server.

`log-file <filename>`

The 'log-file' will open the 'filename' and instruct the server to start writing the debug log into it. Note that ssh-server-ctl will return immediately but the server will continue writing the log until 'clear' command is issued. This command will not affect the current debug level.

`syslog <on|off>`

The 'syslog' command is used to turn on or off writing debug messages to syslog in Unix systems. The default is off. This command will not affect the current debug level.

Note

Enable only if there are handful of connections and low <debug_level> has been set in order to avoid syslogd slowing down the processes and system further. Available on Unix only.

`monitor`

The 'monitor' will cause ssh-server-ctl to stay running and print out the server logs. This command will not affect current debug level. Not supported on SELinux-enabled systems.

Apart from normal debug level <debug_level> definitions such as '4' or 'sshuser*=9,6', the server supports the following SFTP related debug options, that can be used as debug level for 'set' command with 'sftpfile=<filename>;sftpdebug=<level>' syntax.

`sftpfile=<filename>`

Passed to the sft-server-g3, the sft server will write its debug to the given file. The `%U` will be substituted with a user name and the `%H` with a host name in file path.

`sftpdebug=<level>`

Normally the sft-server-g3 will get the same debug level as the ssh-server-g3. With this a different debug level can be used in sft and in main server.

`sftpstderrdebug=<level>`

By default sft debug messages are sent to the ssh-servant-g3 process. This enables sending standard error to the client.

The following example sets the global debug level to 4 for all Tectia Server processes, SFTP debug level to 8 with user-specific logs.

```
ssh-server-ctl debug set '4;sftpfile=/tmp/sftp_%U.log;sftpdebug=8'
```

`performance`

Show and manipulate server performance profiling data.

`enroll-certificate`

Enroll a certificate for user or host authentication.

`fips-mode`

Check or change FIPS mode switch for installed Tectia products.

`hostkey`

Shows the server's hostkey and information.

`pause`

Pause the service. Existing connections continue to function, but new connections will not be accepted until the **continue** command has been given.

`pid`

Prints the server process ID.

`reload`

Causes the server process to validate and reload its configuration. The configuration is read from the `ssh-server-config.xml` file. Existing connections stay open using the old configuration and the new connections will use the new configuration.

`start`

Attempts to start the server process by executing **ssh-server-g3**.

The `start` command will check if there is a server process currently running; if yes, the tool will report the case and will not make any starting attempts. Available on Unix only.

Options:

`-p, --port` *`PORT`*

Start the server on an alternate port (the default port is 22).

`-f, --config-file` *`FILE`*

Uses the given file as a configuration.

`--server-log-file` *`FILE`*

Write server debug log to FILE, for example `/tmp/server-debug.txt`

`--server-debug-level` *`LEVEL`*

Set debug LEVEL to server on startup.

Note

If debug is enabled, remember to disable it with **ssh-server-ctl debug clear** command. It should always be used after reproducing the problem, even if the service will be stopped on Unix. If the service has been restarted since debug mode was enabled on Unix, the **ssh-server-ctl --old debug clear** is needed to clear the debug settings from any retired server processes.

`status`

When the server is running, this command outputs the following information:

- Server status and process ID
- Server version
- Date and time of starting the server
- Cryptographic mode (FIPS/Standard)
- Cryptographic library version
- PQC library version

- Address family type
- Path to the server control socket
- Number of successful reconfigurations
- Date and time of the last reconfiguration (if applicable)
- Number of connections received
- Number of servants
- Preferred number of servants
- Maximum number of servants
- Maximum number of connections per servant
- Total number of connections after which servants are retired
- Hostkey info and SHA-256 fingerprint for current server identity
- Future hostkey info and SHA-256 fingerprint for .next key (if automatic rotation is enabled)
- Debug level if enabled
- Status of load control (enabled/disabled)
- Maximum size for the server's white list
- Discard limit for the server's white list
- Additionally, for each servant:
  - Servant ID number, process ID and status
  - Number of current connections, unauthenticated connections, and channels
  - All-time total number of authenticated connections
  - Current new connections in message queue (with --verbose option)
  - Accepting white-listed connections only (if applicable)
  - Total connections received (with --verbose option)
- Server's listening port number
- Current servant ID number that will receive the next incoming connection (with --verbose option)

`stop`

Causes the server process to start shutting down. The `stop` command checks if there is a server process currently running; if not, the tool will report the case and will not make any stopping attempts.

On AIX, if an error occurs when the server is stopped by using **ssh-server-ctl stop**, it falls back to stop the server process directly.

Options:

`-F, --force`

Forcefully disconnects connections to shut down the server quicker. The `force` option should be given with the initial **stop** command.

`stop-servant` [`command-options`] *`id ...`*

Instructs the server to stop servants specified by their ID numbers. You can use a space-separated list to enter several IDs or `all` in which case all running servants will be stopped. If `one` is specified, a servant with least number of connections is selected and then either stopped or retired, depending on other command-options.

If number of running servants falls below preferred number, a new servant will be started automatically unless --no-restart option is used.

Options:

`--retire`

Retire servants instead of stopping them immediately. The servant stops accepting new connections and will exit after the last client connection disconnects.

`--no-restart`

Do not start new servant, lowers preferred number of servants if needed.

`view-white-list`

Prints the IP addresses on the server's white list in reverse chronological order. The white list is a list of IP addresses of connections that have recently had a successful authentication. (For more information, see **`load-control`** in ssh-server-config(5).)

The following commands related to password cache management are supported on Windows only:

`add-pwd-cache-user` *`username`*

(*Windows only*) Adds the specified user and entered password to the server password cache database. If the user already exists, the existing password gets replaced with the new one. The password is read from the console or standard input.

`del-pwd-cache-user` *`username`*

(*Windows only*) Removes the specified user's password from the currently active server password cache. The command will report an error if the specified user is not present in the password cache.

`export-pwd-cache` [`--password`] *`FILE`*

(*Windows only*) Exports the current password database into an external encrypted *FILE*. If *FILE* already exists, the export will fail (no files will be overwritten). *FILE* must reside on a local drive.

The password that will be used to protect the password database *FILE* will be requested interactively, unless it is provided as a command-line argument (with the `--password` option).

Options:

`--password=` *PASSWORD* | `file://` *PASSWORDFILE* | `env://` *VARIABLE* | `extprog://` *PROGRAM*

Sets the user password. It is possible to give the *PASSWORD* directly as an argument to this option or through an environment *VARIABLE*, but these options are not recommended. Better alternatives are entering a path to a file on a local drive that contains the password (`--password=file://` *PASSWORDFILE*), or entering a path to an external program or script that outputs the password (`--password=extprog://`*PROGRAM*).

Caution

Supplying the password on the command line or setting it as an environment variable are not secure options. For example, in a multiuser environment, the password given directly on the command line is trivial to recover from the process table.

Note

If you use a *PASSWORDFILE*, make sure that its permissions are set so that only the owner of the file has access to it. Note also that the export operation is run within the account in which the Server is running (System) - not within the account that is used to run **ssh-server-ctl**.

Tectia enforces the use of strong passwords for the password cache export and import functions. Instead of explicit password requirements, we use a "password class" system. For example, a password that consists of eight unique characters from three different character classes or a password of eleven unique characters from two character classes are deemed strong enough. The character classes are: digits, lower-case letters, upper-case letters, and other characters. When calculating the number of different character classes, upper-case letters used as the first character and digits used as the last character of a password are ignored.

`import-pwd-cache` [`--password`] *FILE*

(*Windows only*) Imports a previously exported password database from an external encrypted *FILE*. The external password database *FILE* must reside on a local drive. All entries from *FILE* will be added or overwritten into the current password cache database that is in use by the server. The password that is protecting the *FILE* will be requested interactively, unless it is provided as a command-line argument (with the `--password` option).

Caution

The passwords of user names that already exist in the current password cache will be overwritten by those in the imported password database *FILE*.

Options are the same as for **export-pwd-cache**.

`show-pwd-cache-users`

(*Windows only*) Displays all stored user names from the currently active server password cache. The passwords are not displayed, only the user names.

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 Tectia installation (installed product components and versions, their state, and the global and user-specific configurations).

The collected information will be stored in a file named `ssh_troubleshoot_<host>_<date>_<time>.tar` on Unix and `ssh_troubleshoot_*.log` on Windows. Send the file to the SSH technical support for analysis to help in troubleshooting situations.

To get all necessary information, run the command as an administrator, because it might need root access to some directories.

Options

Enter each option separately, they cannot be combined. The following options are available:

`-d, --debug` *`LEVEL`*

Sets the debug level string to *`LEVEL`*.

`-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 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.

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. This tool can also convert openSSH public or private keys to the Tectia key format, or, from Tectia key format to openSSH format. Tectia public keys use The Secure Shell (SSH) Public Key File Format (RFC 4716).

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 file name is specified, the key pair is likewise stored under the user's home directory with such file names as `id_type_bits_a` and `id_type_bits_a.pub`.

When specifying file paths or other strings that contain spaces, enclose them in quotation marks ("").

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. The allowed and default lengths for different key types are:

- RSA or DSA: allowed 512 to 65536 bits, default 3072 bits
- ECDSA: allowed 256, 384 and 521 bits, default 384 bits
- Ed25519: allowed/default 256 bits

`-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 and type (RSA, DSA, ECDSA or Ed25519) 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 behavior of this option: `--fingerprint-type` `--hash`, `--hostkeys-directory`, `--known-hosts`, `--rfc4716`.

`-F, --fingerprint host_ID`

Dumps the location, fingerprint and type (RSA, DSA, ECDSA or Ed25519) of the locally stored host key(s) 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 behavior 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, "`<INSTALLDIR>\SSH Tectia Server`" on Windows). By default stores the private key with an empty passphrase or if in FIPS mode with random passphrase in `<privatekey>.pass`. Use `--prompt-pass` option to prompt for the passphrase in `--hostkey` mode.

`-i file`

Loads and displays information on the key `file`.

`--pass-file file`

Read passphrase from a `file` for displaying *-i* information on a passphrase protected private key. If *<privatekey>.pass* exists, it is used by default.

`-p` *`passphrase`*

Specifies the passphrase for the generated key.

`-P`

Specifies that the generated key will be saved with an empty passphrase.

Note

In FIPS mode, due to a FIPS regulation which forbids exporting unencrypted private keys out of the FIPS module, it is not possible to generate user keys without a passphrase.

`--random-pass`

Create random passphrase for the generated private key and save it to *<privatekey>.pass*. By default the passprase is Base64 encoded.

`-k` *`file`*

Converts a PKCS #12 file to an SSH2-format certificate and private key.

`-m, --generate-moduli-file`

Generates moduli file `secsh_dh_gex_moduli` for Diffie-Hellman group exchange.

`-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` | `ecdsa` | `ed25519`

Selects the type of the key. Valid values are `rsa` (default), `dsa`, `ecdsa`, and `ed25519`.

`-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 behavior 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 behavior of this option: `--host-key-file`, `--hostkeys-directory`, `--known-hosts`.

`--hash` `sha256` | `sha1` | `md5`

Specifies the digest algorithm for fingerprint generation. Valid options are `sha256`, `sha1` and `md5`. The default is sha1.

`--fingerprint-type` `base64` | `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 file name 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.

The keys must have non-empty passphrases.

By default (if this option is not given or Tectia FIPSMODE switch file is not present), 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.

`--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 the format specified by --key-format parameter. If *`outfile`* is not given, it will be requested. The default output format is SSH2 native format.

`--import-private-key` *`infile`* [*`outfile`*]

Attempts to import a private key from *`infile`* and store it to *`outfile`* in the format specified by --key-format parameter. If *`outfile`* is not given, it will be requested. The default output format is 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`*.

`--key-format` *`format`*

Output key format: secsh2, pkcs1, pkcs8, pkcs12, openssh2, or openssh2-aes.

`--key-hash` *`hash`*

This option can be used for other than Tectia key formats. Specifies the hash algorithm to be used in passphrase-based private key derivation. The default value is `sha1`. Other supported algorithms are `sha224`, `sha256`, `sha384`, and `sha512`. Note that all key formats do not support all hash algorithms.

`--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.

`--moduli-file-name` *`file`*

Writes the moduli generated for Diffie-Hellman group exchange to *`file`*. (The default file name for option `-m` is `secsh_dh_gex_moduli`.)

`--overwrite` [`=yes` | `no`]

Overwrite files with the same file names. 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 Tectia Server installation.

`--sign-cert` *`file`*

Make a certificate with the generated public key, and write to *`file`*. For a complete list of additional certificate options, view the option help with **--sign-cert help**.

`-V`

Displays version string and exits.

`-h, --help, -?`

Displays a short summary of command-line options and exits.

Examples

Create a 3072-bit RSA key pair using the cryptographic library in the FIPS mode and store the key pair in the default user key directory with file names *`newkey`* and *`newkey.pub`*:

```
$ ssh-keygen-g3 --fips-mode -b 3072 newkey
```

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 Base64-encoded SHA256 fingerprint of the public hostkey:

```
$ ssh-keygen-g3 --hash sha256 --fingerprint-type base64 -F hostkey.pub
Fingerprint for key `hostkey.pub':
9UmbXHpUodKPXS0pFIACGLjKoiHQBShPVZj6ShUNWgM (RSA)
```

Convert a private key into openSSH2-AES format:

```
$ ssh-keygen-g3 -p <password> --key-format openssh2-aes \
  --import-private-key <source_key_file> <destination_key_file>
```

Note: if the private key file that is being converted is encrypted with a passphrase, the passphrase must be provided with the '-p' option.

Convert a Tectia public key `tectiakey.pub` to an OpenSSH public key `opensshkey.pub`:

```
$ ssh-keygen-g3 --key-format openssh2 --import-public-key \
  tectiakey.pub opensshkey.pub
```

Generate moduli file `dhgex-moduli` for Diffie-Hellman group exchange:

```
$ ssh-keygen-g3 -m --moduli-file-name dhgex-moduli
```

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 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-syxyx
```

Accept the server key as a known key for Tectia Server 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-nybuv-muled-syxyx-xigad-hozuf-kykek-vogid-dumid
```

Accept the server key as a known key for 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-nybuv-muled-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-nybuv-muled-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 given 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.
     -d level      Set debug level.
     -Z provspec   Specifies external key provider for the private key.
                   The format of provspec is "providername:initstring".


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 (Tectia-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 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 specifying subject names or other strings that contain spaces, enclose them in quotation marks ("").

`-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.

`-d level`

Sets the debug level string to *`level`*.

`-Z provspec`

Specifies the external key provider for the private key. Give *`provspec`* in the format *`"providername:initstring"`*.

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`, `nonRepudiation`, `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:2048/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:2048/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  (Tectia proprietary crypto library format, passphrase-protected)
 - pkcs1 (PKCS#1 format)
 - pkcs8s (passphrase-protected PKCS#8, "shrouded PKCS#8")
 - pkcs8 (plain-text PKCS#8)
 - x509 (Tectia 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 `.crt` 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`, `nonRepudiation`, `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-ca1.ssh.com`.

```
$ ssh-scepclient-g3 GET-CA \
   -o ca http://pki.ssh.com:8080/scep/ \
   test-ca1.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:2048/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) :
        96356809228059302634765496419579987563410225412029378652405 53
        93747409460794737674242240714708377288408393205216215183233 77
```

```
      359310235041598725230081792676996888115989695549027436860666
      4
      075964413169075053266526621869646606037779935803673547590225
      7
      608609856291936396347092669016274425845198312457559592684955
      1
      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. 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.

Appendix D Audit Messages

This appendix lists the audit messages generated by the server.

100 Server_starting
Level: notice
Origin: Tectia Server

The server is starting.

Default log facility: daemon

101 Server_start_failed
Level: error
Origin: Tectia Server

The server encountered an unrecoverable error during the startup.

Default log facility: daemon

Argument	Description
Success | Error	Error code
Text	Verbose description of the error

102 Server_running
Level: notice
Origin: Tectia Server

The server has started and is running normally.

Default log facility: daemon

Argument	Description
Ver	Server version string.

Argument	Description
Text	Cryptolibrary version string.

103 Server_stopping
Level: notice
Origin: Tectia Server

The server is shutting down.

Default log facility: daemon

104 Server_exiting
Level: notice
Origin: Tectia Server

The server is exiting.

Default log facility: daemon

105 Server_reconfig_started
Level: informational
Origin: Tectia Server

The server is starting the reconfiguration operation

Default log facility: daemon

Argument	Description
File name	The name of the configuration file

106 Server_reconfig_finished
Level: notice
Origin: Tectia Server

The server reconfiguration has finished.

Default log facility: daemon

Argument	Description
Success | Error	"Success" or short error description.
Text	Verbose description of the error.

107 Server_listener_started
Level: notice
Origin: Tectia Server

The server successfully established a listener socket.

Default log facility: daemon

Argument	Description
Listener	Address of the listener socket

Argument	Description
Listener Port	Port of the listener socket

108 Server_listener_stopped
Level: notice
Origin: Tectia Server

The server closed a listener socket.

Default log facility: daemon

Argument	Description
Listener	Address of the listener socket
Listener Port	Port of the listener socket

109 Server_listener_failed
Level: warning
Origin: Tectia Server

The server failed to open a listener socket.

Default log facility: daemon

Argument	Description
Listener	Address of the listener socket
Listener Port	Port of the listener socket

110 Servant_exited
Level: warning
Origin: Tectia Server

A servant has exited, possibly because of an error.

Default log facility: daemon

Argument	Description
Pid	Process ID of the servant
Success | Error	Success or error code
Exit Value	Exit value of the servant process

111 Servant_error
Level: warning
Origin: Tectia Server

A servant reported an unrecoverable error

Default log facility: daemon

Argument	Description
Pid	Process ID of the servant
Text	Textual error message

112 Server_error
Level: warning
Origin: Tectia Server

The crypto library FIPS status was changed in reconfiguration.

Default log facility: daemon

Argument	Description
Text	Verbose description of the error

113 Server_warning
Level: warning
Origin: Tectia Server

The server encountered a non-fatal error.

Default log facility: daemon

Argument	Description
Text	Description of the error

114 Servant_warning
Level: warning
Origin: Tectia Server

The server encountered a non-fatal error condition.

Default log facility: daemon

Argument	Description
Text	Verbose warning message
Session-Id	Session identifier, if available

115 Servant_info
Level: informational
Origin: Tectia Server

The server encountered a non-error condition of interest.

Default log facility: daemon

Argument	Description
Text	Verbose informational message
Session-Id	Session identifier, if available

116 Servant_client_verbose
Level: notice
Origin: Tectia Server

The Client sent a high priority debug message.

Default log facility: daemon

Argument	Description
Text	Client debug message
Session-Id	Session identifier

117 Servant_client_debug
Level: informational
Origin: Tectia Server

The Client sent a low priority debug message.

Default log facility: daemon

Argument	Description
Text	Client debug message
Session-Id	Session identifier

126 Password_cache_export_started
Level: informational
Origin: Tectia Server

Password cache export started.

Default log facility: daemon

Argument	Description
File name	Name of the file into which the password database will be exported

127 Password_cache_export_finished
Level: informational
Origin: Tectia Server

Password cache export finished.

Default log facility: daemon

Argument	Description
File name	Name of the file into which the password database was exported
Status	Result of the export operation
Text	Optional details reported in case of an error

128 Password_cache_import_started
Level: informational
Origin: Tectia Server

Password cache export started.

Default log facility: daemon

Argument	Description
File name	Name of the file from which the password database will be imported

129 Password_cache_import_finished
Level: informational
Origin: Tectia Server

Password cache export finished.

Default log facility: daemon

Argument	Description
File name	Name of the file from which the password database was imported
Status	Result of import operation
Text	Optional details reported in case of an error

130 Server_hostkey_rotation_started
Level: informational
Origin: Tectia Server

Hostkey rotation started by creating new hostkey. The new key will be advertised during the renewal period while the old key is still used to authenticate the server. After renewal period the new key will become the hostkey.

Default log facility: daemon

Argument	Description
Text	Description of the event.
Text	Time when the hostkey will be rotated.
Text	SHA-1 fingerprint of the new key.
Text	SHA-256 key digest

131 Server_hostkey_rotation
Level: informational
Origin: Tectia Server

The old hostkey is now replaced with the previously created and advertised key. This concludes the automatic hostkey rotation. Clients which have not been able to update their known hosts until now will not be able to connect anymore.

Default log facility: daemon

Argument	Description
Text	Description of the event.
Text	Next rotation time.
Text	Next new hostkey generated at
Text	SHA-1 fingerprint of the hostkey.

Argument	Description
Text	SHA-256 key digest

132 Hostkey_advert_accepted
Level: informational
Origin: Tectia Server

The Client accepted new hostkey and requested server to prove ownership.

Default log facility: daemon

Argument	Description
Username	User's login name.
Src	Remote hostname
Src IP	Remote IP address
Src Port	Remote port
Text	SHA-1 fingerprint of the new hostkey.
Text	SHA-256 key digest.
Session-Id	Session identifier.
Protocol-session-Id	Protocol session identifier.

140 Server_hostkey
Level: informational
Origin: Tectia Server, Connection Broker

Server hostkey properties.

Default log facility: normal

Argument	Description
Status	Key failed.
Text	Key file name
Text	Key type.
Public key hash	Public key hash (MD5).
Public key hash	Public key hash (SHA-1).
Public key hash (SHA-256)	Public key hash (SHA-256).
Text	Error message.

400 Connect
Level: security-success
Origin: Tectia Server

The server initially accepted an incoming connection.

Default log facility: normal

Argument	Description
Policy name	Symbolic name for the connection policy definition in the server configuration (optional)

Argument	Description
Src	Remote hostname
Src IP	Remote IP address
Dst IFace	Local interface ID
Dst IP	Local IP address
Src Port	Remote port
Dst Port	Local port
Ver	Client's version string
Session-Id	Session id

401 Connection_denied
Level: security-failure
Origin: Tectia Server

Connection was denied because maximum number of connections was reached.

Default log facility: normal

Argument	Description
Success | Error	Reason for the connection being denied
Src IP	Remote IP address
Dst IFace	Local interface ID
Dst IP	Local IP address
Src Port	Remote port
Dst Port	Local port.
Text	Textual information of the disconnect reason.

402 Disconnect
Level: informational
Origin: Tectia Server

The connection has closed.

Default log facility: normal

Argument	Description
Reason	Reason for disconnect
Src	Remote hostname
Src IP	Remote IP address
Dst IFace	Local interface ID
Dst IP	Local IP address
Src Port	Remote port
Dst Port	Local port
Text	Textual description of the disconnect reason
Session-Id	Session identifier

403 Version_exchange_failure
Level: security-failure

Origin: Tectia Server

Connection was denied because an error occurred during the version negotiation.

Default log facility: normal

Argument	Description
Success | Error	Reason for the denied connection.
Src IP	Remote IP address
Dst IFace	Local interface ID
Dst IP	Local IP address
Src Port	Remote port
Dst Port	Local port.
Text	Textual information of the disconnect reason.
Session-Id	Session identifier

410 Login_success
Level: security-success
Origin: Tectia Server

The user has logged in after successful authentication, account validity and other checks.

Default log facility: normal

Argument	Description
Username	The user's login name
Src	Remote hostname
Src IP	Remote IP address
Dst IFace	Local interface ID
Dst IP	Local IP address
Src Port	Remote port
Dst Port	Local port
Ver	Client's version string
Session-Id	Session identifier

411 Login_failure
Level: security-failure
Origin: Tectia Server

An unsuccessful login attempt was made to the server.

Default log facility: normal

Argument	Description
Username	The user's login name
Reason	Reason for disconnect
Src	Remote hostname
Src IP	Remote IP address
Dst IFace	Local interface ID

Argument	Description
Dst IP	Local IP address
Src Port	Remote port
Dst Port	Local port
Text	Textual description of the disconnect reason
Text	Connection error messages
Session-Id	Session identifier

412 Logout
Level: informational
Origin: Tectia Server

The user has logged out.

Default log facility: normal

Argument	Description
Username	The user's login name
Reason	Reason for disconnect
Src	Remote hostname
Src IP	Remote IP address
Dst IFace	Local interface ID
Dst IP	Local IP address
Src Port	Remote port
Dst Port	Local port
Text	Textual description of the disconnect reason
Session-Id	Session identifier

420 Session_channel_open
Level: informational
Origin: Tectia Server

A session channel opening has completed successfully or unsuccessfully.

Default log facility: normal

Argument	Description
Username	User's login name
Success | Error	Success or error description
Command	The requested command
Sub ID	Channel number
Session-Id	Session identifier

421 Session_channel_close
Level: informational
Origin: Tectia Server

A session channel has closed.

Default log facility: normal

Argument	Description
Username	User's login name
Sub ID	Channel number
Session-Id	Session identifier

422 Forwarding_channel_open
Level: informational
Origin: Tectia Server

A TCP forwarding channel request to the client has completed successfully or unsuccessfully.

Default log facility: normal

Argument	Description
Username	User's login name
Success | Error	Success or error description
Tunnel type	"Server to client"
Src IP	Originator's IP address
Src Port	Originator's port
Listener IP	Receiving listener address
Listener Port	Receiving listener port
Sub ID	Channel number
Session-Id	Session identifier

423 Forwarding_channel_close
Level: informational
Origin: Tectia Server

A TCP forwarding channel has closed.

Default log facility: normal

Argument	Description
Username	User's login name
Sub ID	Channel number
Session-Id	Session identifier

424 Auth_channel_open
Level: informational
Origin: Tectia Server

An authorization agent forwarding channel opening has completed successfully or unsuccessfully.

Default log facility: normal

Argument	Description
Username	User's login name

Argument	Description
Sub ID	Channel number
File name	Path to the listener
Success | Error	Success or error description
Session-Id	Session identifier

425 Auth_channel_close
Level: informational
Origin: Tectia Server

An authorization agent forwarding channel has closed.

Default log facility: normal

Argument	Description
Username	User's login name
Sub ID	Channel number
File name	Path to the listener
Session-Id	Session identifier

426 Forwarding_listener_open
Level: informational
Origin: Tectia Server

A TCP forwarding listener opening has completed successfully or unsuccessfully.

Default log facility: normal

Argument	Description
Username	User's login name
Success | Error	Success or error description
Listener IP	Listener's IP address
Listener Port	Listener's port
Session-Id	Session identifier
Text	Error description

427 Forwarding_listener_close
Level: informational
Origin: Tectia Server

A TCP forwarding listener has been closed.

Default log facility: normal

Argument	Description
Username	User's login name
Listener IP	Listener's IP address
Listener Port	Listener's port
Session-Id	Session identifier

428 Auth_listener_open
Level: informational
Origin: Tectia Server

An authorization agent forwarding listener opening has completed successfully or unsuccessfully.

Default log facility: normal

Argument	Description
Username	User's login name
File name	Path to the listener
Success | Error	Success or error description
Session-Id	Session identifier

429 Auth_listener_close
Level: informational
Origin: Tectia Server

An authorization agent forwarding listener has closed.

Default log facility: normal

Argument	Description
Username	User's login name
File name	Path to the listener
Session-Id	Session identifier

430 Channel_open_failure
Level: informational
Origin: Tectia Server

Opening a channel has failed, because the other end has returned a failure.

Default log facility: normal

Argument	Description
Username	User's login name
Sub ID	Channel number
Failure code	Error code from the other end
Text	Textual description of the error from the other end.
Session-Id	Session identifier

431 X11_listener_open
Level: informational
Origin: Tectia Server

An X11 forwarding listener opening has completed successfully or unsuccessfully.

Default log facility: normal

Argument	Description
Username	User's login name
Display	Display name
Success | Error	Success or error description
Text	Textual description of the error (optional)
Session-Id	Session identifier

432 X11_listener_close
Level: informational
Origin: Tectia Server

An X11 forwarding listener has closed.

Default log facility: normal

Argument	Description
Username	User's login name
Display	Display name
Session-Id	Session identifier

433 X11_channel_open
Level: informational
Origin: Tectia Server

An X11 forwarding channel opening has completed successfully or unsuccessfully.

Default log facility: normal

Argument	Description
Username	User's login name
Sub ID	Channel number
Display	Display name
Success | Error	Success or error description
Text	Extra textual information (optional)
Session-Id	Session identifier

434 X11_channel_close
Level: informational
Origin: Tectia Server

An X11 forwarding channel has closed.

Default log facility: normal

Argument	Description
Username	User's login name
Sub ID	Channel number
Session-Id	Session identifier

460 Session_env_request_failure

Level: informational
Origin: Tectia Server

A session channel request to set an environment variable has failed.

Default log facility: normal

Argument	Description
Username	User's login name
Success | Error	Success or error description
Text	Description of the error
Sub ID	Channel number
Session-Id	Session identifier

461 Session_env_file_failure
Level: informational
Origin: Tectia Server

When reading the user's environment file, an illegal environment variable was encountered.

Default log facility: normal

Argument	Description
Username	User's login name
Success | Error	Success or error description
File name	Environment file name
Text	Description of the error
Session-Id	Session identifier

700 Auth_method_success
Level: informational
Origin: Tectia Server

An authentication method was successfully completed.

Default log facility: auth

Argument	Description
Username	User's login name
Auth method	Authentication method's name
Session-Id	Session identifier

701 Auth_method_failure
Level: informational
Origin: Tectia Server

An authentication method failed.

Default log facility: auth

Argument	Description
Username	User's login name
Auth method	Authentication method's name
Session-Id	Session identifier

702 Auth_methods_completed
Level: informational
Origin: Tectia Server

Authentication was successful. Display authentication methods and client details.

Default log facility: auth

Argument	Description
Username	User's login name
Auth methods	Completed auth methods
Src IP	Client IP address
Src Port	Client port
Ver	Client protocol version string
Session-Id	Session identifier.

703 Auth_methods_available
Level: informational
Origin: Tectia Server

The list of authentication methods still available for the user

Default log facility: auth

Argument	Description
Username	User's login name
Auth methods	A list of available authentication methods
Session-Id	Session identifier

705 Auth_error
Level: warning
Origin: Tectia Server

An error occurred during authentication.

Default log facility: auth

Argument	Description
Username	User's login name
Text	Textual description of the error
Session-Id	Session identifier

706 Auth_warning
Level: informational
Origin: Tectia Server

Something abnormal happened during authentication, but authentication can still continue.

Default log facility: auth

Argument	Description
Username	User's login name
Text	Textual description of the error
Session-Id	Session identifier

707 Publickey_auth_success
Level: informational
Origin: Tectia Server

The user successfully completed the publickey authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'publickey'
Text	Acceptance description, includes key fingerprint
Session-Id	Session identifier

708 Publickey_auth_error
Level: warning
Origin: Tectia Server

Error in the publickey authentication method. This means that the current user authentication attempt with the publickey method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'publickey'
Text	Error description
Session-Id	Session identifier

709 Publickey_auth_warning
Level: informational
Origin: Tectia Server

Warning during the publickey authentication method. This means that something abnormal happened during the publickey method, but the method may still complete successfully.

Default log facility: auth

Argument	Description
Username	User's login name

Argument	Description
Algorithm	'publickey'
Text	Warning description
Session-Id	Session identifier

711 Hostbased_auth_error
Level: warning
Origin: Tectia Server

Error in the hostbased authentication method. This means that the current user authentication attempt with the hostbased method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

712 Hostbased_auth_warning
Level: informational
Origin: Tectia Server

Warning during the hostbased authentication method. This means that something abnormal happened during the hostbased authentication method, but the method may still complete successfully.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Warning description
Session-Id	Session identifier

714 Password_auth_error
Level: warning
Origin: Tectia Server

Error in the password authentication method. This means that the current user authentication attempt with the password method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'password'
Text	Error description
Session-Id	Session identifier

715 Password_auth_warning
Level: informational
Origin: Tectia Server

Warning during the password authentication method. This means that something abnormal happened in the password method, but the method may still complete successfully.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'password'
Text	Warning description
Session-Id	Session identifier

716 Keyboard_interactive_pam_auth_success
Level: informational
Origin: Tectia Server

The user successfully completed the password PAM authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'password'
Text	Acceptance description
Session-Id	Session identifier

717 Keyboard_interactive_pam_auth_error
Level: warning
Origin: Tectia Server

Error in the password PAM authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

718 Keyboard_interactive_pam_auth_warning
Level: informational
Origin: Tectia Server

Warning during the keyboard-interactive PAM authentication method. This means that something abnormal happened, but the method may still complete successfully.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

719 Keyboard_interactive_radius_auth_success
Level: informational
Origin: Tectia Server

The user successfully completed the Keyboard-interactive Radius authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'Keyboard-interactive Radius'
Text	Acceptance description
Session-Id	Session identifier

720 Keyboard_interactive_radius_auth_error
Level: warning
Origin: Tectia Server

Error in the Keyboard-interactive Radius authentication method. This means that the current user authentication attempt with the Keyboard-interactive Radius method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

722 Keyboard_interactive_password_auth_success
Level: informational
Origin: Tectia Server

The user successfully completed the Keyboard-interactive Password authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'Keyboard-interactive Password'
Text	Acceptance description

Argument	Description
Session-Id	Session identifier

723 Keyboard_interactive_password_auth_error
Level: warning
Origin: Tectia Server

Error in the Keyboard-interactive password authentication method. This means that the current user authentication attempt with the Keyboard-interactive password method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

725 Keyboard_interactive_securid_auth_success
Level: informational
Origin: Tectia Server

The user successfully completed the Keyboard-interactive SecurID authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'Keyboard-interactive SecurID'
Text	Acceptance description
Session-Id	Session identifier

726 Keyboard_interactive_securid_auth_error
Level: warning
Origin: Tectia Server

Error in the Keyboard-interactive SecurID authentication method. This means that the current user authentication attempt with the Keyboard-interactive SecurID method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

727 Keyboard_interactive_securid_auth_warning
Level: informational

Origin: Tectia Server

Warning during the Keyboard-interactive SecurID authentication method. This means that something abnormal happened during the Keyboard-interactive SecurID authentication method, but the method may still complete successfully.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Warning description
Session-Id	Session identifier

728 GSSAPI_auth_success
Level: informational
Origin: Tectia Server

The user successfully completed the GSSAPI authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'GSSAPI'
Text	Acceptance description
Session-Id	Session identifier

729 GSSAPI_auth_error
Level: warning
Origin: Tectia Server

Error in the GSSAPI authentication method. This means that the current user authentication attempt with the GSSAPI method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

730 GSSAPI_auth_warning
Level: informational
Origin: Tectia Server

Warning during the GSSAPI authentication method. This means that something abnormal happened during the GSSAPI authentication method, but the method may still complete successfully.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Warning description
Session-Id	Session identifier

731 Keyboard_interactive_aix_lam_auth_success
Level: informational
Origin: Tectia Server

The user successfully completed the Keyboard-interactive AIX LAM authentication method.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	'Keyboard-interactive AIX LAM'
Text	Acceptance description
Session-Id	Session identifier

732 Keyboard_interactive_aix_lam_auth_error
Level: warning
Origin: Tectia Server

Error in the Keyboard-interactive AIX LAM authentication method. This means that the current user authentication attempt with the Keyboard-interactive AIX LAM method has failed.

Default log facility: auth

Argument	Description
Username	User's login name
Algorithm	Authentication method name
Text	Error description
Session-Id	Session identifier

800 Rule_engine_failure
Level: error
Origin: Tectia Server

An internal error occurred in rule engine.

Default log facility: normal

Argument	Description
Username	The user's login name, if available.
Policy name	Current authentication- or rule-element's symbolic name, if available.
Text	Textual representation of the error.

Argument	Description
Session-Id	Session id, if available.
Text	Authentication block line information.

801 Authentication_block_selected
Level: informational
Origin: Tectia Server

The server selected an authentication block in the configuration.

Default log facility: normal

Argument	Description
Username	The user's login name
Policy name	A symbolic name for the authentication block
Session-Id	Session identifier
Text	Authentication block line information.

802 Authentication_block_allow
Level: informational
Origin: Tectia Server

The authentication chain terminated into an authentication block with an allow action.

Default log facility: normal

Argument	Description
Username	The user's login name
Policy name	A symbolic name for the authentication block
Session-Id	Session identifier
Text	Authentication block line information.

803 Authentication_block_deny
Level: informational
Origin: Tectia Server

The authentication chain terminated into an authentication block with a deny action.

Default log facility: normal

Argument	Description
Username	The user's login name
Policy name	A symbolic name for the authentication block
Session-Id	Session identifier
Text	Authentication block line information.

804 Group_selected
Level: informational
Origin: Tectia Server

The connection matched a group in the configuration.

Default log facility: normal

Argument	Description
Username	The user's login name
Policy name	A symbolic name for the group
Session-Id	Session identifier

805 Rule_selected
Level: informational
Origin: Tectia Server

The server selected a rule in the configuration.

Default log facility: normal

Argument	Description
Username	The user's login name
Policy name	A symbolic name for the rule
Session-Id	Session identifier
Text	Authentication block line information.

806 Rule_engine_warning
Level: warning
Origin: Tectia Server

A non-fatal error has occurred in the rule engine.

Default log facility: normal

Argument	Description
Text	The warning message.

1000 KEX_failure
Level: warning
Origin: 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: 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: 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: Tectia Server

Key-exchange was successful.

Default log facility: normal

Argument	Description
Algorithm	Kex method name.
Session-Id	Session identifier.
Protocol-session-Id	Protocol session identifier.

1100 Certificate_validation_failure
Level: informational
Origin: 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)

Argument	Description
Text	Resulting search states for all configured CAs.
Session-Id	Session identifier
Text	X.509 certificate subject name.
Text	X.509 certificate serial number.
Text	X.509 certificate email altnames.
Text	X.509 certificate UPN alternative names.

1101 Certificate_validation_success
Level: informational
Origin: 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
Text	X.509 certificate subject name.
Text	X.509 certificate serial number.
Text	X.509 certificate email altnames.
Text	X.509 certificate UPN alternative names.

1110 CM_find_started
Level: informational
Origin: 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: Tectia Server, Connection Broker

A search was completed with a trace of sources used.

Default log facility: normal

Argument	Description
Ctx	The context pointer identifying the search
Text	Search trace identifiying source used.

1112 CM_cert_not_in_search_interval
Level: informational
Origin: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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

1200 Key_store_create
Level: informational
Origin: Tectia Server, Connection Broker

Key store created.

Default log facility: normal

1201 Key_store_create_failed
Level: warning
Origin: Tectia Server, Connection Broker

Key store creation failed.

Default log facility: normal

1202 Key_store_destroy
Level: informational
Origin: Tectia Server, Connection Broker

Key store destroyed.

Default log facility: normal

1204 Key_store_add_provider
Level: informational
Origin: Tectia Server, Connection Broker

Added a provider to the key store.

Default log facility: normal

Argument	Description
Type	Provider type

1205 Key_store_add_provider_failed
Level: warning

Origin: Tectia Server, Connection Broker

Adding a provider to the key store failed.

Default log facility: normal

Argument	Description
Type	Provider type
EK error	Error message

1206 Key_store_remove_provider
Level: informational
Origin: Tectia Server, Connection Broker

Removed a provider from the key store.

Default log facility: normal

Argument	Description
Init info	Provider name

1208 Key_store_decrypt
Level: informational
Origin: Tectia Server, Connection Broker

A key was used successfully for decryption.

Default log facility: normal

Argument	Description
Key path	Key path
Fwd path	Fwd path

1209 Key_store_decrypt_failed
Level: warning
Origin: 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: 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
Public key hash	Public key hash (SHA-1)
Public key hash (SHA-256)	Public key hash (SHA-256)

1211 Key_store_sign_failed
Level: warning
Origin: 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
Public key hash	Public key hash (SHA-1)
Public key hash (SHA-256)	Public key hash (SHA-256)

1212 Key_store_sign_digest
Level: informational
Origin: 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
Public key hash	Public key hash (SHA-1)
Public key hash (SHA-256)	Public key hash (SHA-256)

1213 Key_store_sign_digest_failed
Level: warning
Origin: 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

Argument	Description
Public key hash	Public key hash (SHA-1)
Public key hash (SHA-256)	Public key hash (SHA-256)

1214 Key_store_ek_provider_failure
Level: warning
Origin: Tectia Server, Connection Broker

External key provider failure.

Default log facility: normal

Argument	Description
Key path	Key path
Text	Key label
Text	Error description

1220 Key_store_certificate_issued
Level: informational
Origin: Tectia Server, Connection Broker

Internal CA issued a X.509 certificate.

Default log facility: normal

Argument	Description
Text	CA name
Text	Principal name.
Text	Expiration date.
Text	SHA-256 hash of the certificate.

1221 Key_store_certificate_revoked
Level: informational
Origin: Tectia Server, Connection Broker

Internal CA revoked a certificate.

Default log facility: normal

Argument	Description
Text	CA name
Text	Principal name.
Text	Expiration date.
Text	SHA-256 hash of the certificate.

1300 Channel_inbound_statistics
Level: informational
Origin: Connection Broker, 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, 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

2000 Sft_server_starting
Level: notice
Origin: Tectia Secure File Transfer

The SFT server is starting.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Session-Id	Session ID

2001 Sft_server_stopping
Level: notice
Origin: Tectia Secure File Transfer

The SFT server is stopping.

Default log facility: daemon

Argument	Description
Username	User's login name

Argument	Description
Subsystem	Subsystem type
Pid	Process ID of the user process
Session-Id	Session ID

2002 Sft_server_connected
Level: informational
Origin: Tectia Secure File Transfer

The connection to SFT server was accepted.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Session-Id	Session ID

2003 Sft_server_disconnected
Level: informational
Origin: Tectia Secure File Transfer

The connection to SFT server was disconnected.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Session-Id	Session ID

2004 Sft_server_fxp_request
Level: informational
Origin: Tectia Secure File Transfer

Invalid SFT protocol message was received.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Request type	Request type
Request ID	Request ID
Text	Error message
Session-Id	Session ID

2005 Sft_server_fxp_unknown
Level: notice
Origin: Tectia Secure File Transfer

Unknown SFT protocol message was received.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Request type	Request type
Request ID	Request id
Session-Id	Session ID

2006 Sft_server_fxp_version
Level: notice
Origin: Tectia Secure File Transfer

SFT protocol version was negotiated.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Negotiated version	Negotiated protocol version
Session-Id	Session ID

2007 Sft_server_open_file
Level: notice
Origin: Tectia Secure File Transfer

File OPEN was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2008 Sft_server_open_file_failed

Level: warning
Origin: Tectia Secure File Transfer

File OPEN failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2010 Sft_server_read_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File READ failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2012 Sft_server_write_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File WRITE failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2013 Sft_server_close_file
Level: informational
Origin: Tectia Secure File Transfer

File CLOSE was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2014 Sft_server_close_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File CLOSE failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2015 Sft_server_stat_file
Level: informational
Origin: Tectia Secure File Transfer

File STAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message

Argument	Description
Session-Id	Session ID

2016 Sft_server_stat_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File STAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2017 Sft_server_lstat_file
Level: informational
Origin: Tectia Secure File Transfer

File LSTAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2018 Sft_server_lstat_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File LSTAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message

Argument	Description
Session-Id	Session ID

2019 Sft_server_fstat_file
Level: informational
Origin: Tectia Secure File Transfer

File FSTAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2020 Sft_server_fstat_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File FSTAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2021 Sft_server_setstat_file
Level: notice
Origin: Tectia Secure File Transfer

File SETSTAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2022 Sft_server_setstat_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File SETSTAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2023 Sft_server_fsetstat_file
Level: notice
Origin: Tectia Secure File Transfer

File FSETSTAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2024 Sft_server_fsetstat_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File FSETSTAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type

Argument	Description
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2025 Sft_server_remove_file
Level: notice
Origin: Tectia Secure File Transfer

File REMOVE was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2026 Sft_server_remove_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File REMOVE failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2027 Sft_server_rename_file
Level: notice
Origin: Tectia Secure File Transfer

File was renamed.

Default log facility: daemon

Argument	Description
Username	User's login name

Argument	Description
Subsystem	Subsystem type
Pid	Process ID of the user process
Old file name	Old file name
New file name	New file name
Text	Audit message
Session-Id	Session ID

2028 Sft_server_rename_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File RENAME failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2029 Sft_server_symlink
Level: informational
Origin: Tectia Secure File Transfer

Symlink was created.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Link name	Link name
Target name	Target name
Text	Audit message
Session-Id	Session ID

2030 Sft_server_symlink_failed
Level: warning
Origin: Tectia Secure File Transfer

File/directory SYMLINK failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2031 Sft_server_readlink
Level: informational
Origin: Tectia Secure File Transfer

Symlink was resolved.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Link name	Link name
Target name	Resolved target name
Text	Audit message
Session-Id	Session ID

2032 Sft_server_readlink_failed
Level: warning
Origin: Tectia Secure File Transfer

File/directory READLINK failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2033 Sft_server_realpath
Level: informational
Origin: Tectia Secure File Transfer

File name was resolved.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Path	Resolved file name
Text	Audit message
Session-Id	Session ID

2034 Sft_server_realpath_failed
Level: warning
Origin: Tectia Secure File Transfer

File/directory REALPATH failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2035 Sft_server_digest
Level: informational
Origin: Tectia Secure File Transfer

SFT server calculated digest.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Offset	File offset
Length	Length of the file data for digest calculation
Digest	Calculated digest
Session-Id	Session ID

2036 Sft_server_digest_failed
Level: warning
Origin: Tectia Secure File Transfer

File DIGEST failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2037 Sft_server_open_dir
Level: notice
Origin: Tectia Secure File Transfer

Directory OPEN was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2038 Sft_server_open_dir_failed
Level: warning
Origin: Tectia Secure File Transfer

Directory OPEN failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2039 Sft_server_close_dir
Level: informational
Origin: Tectia Secure File Transfer

Directory CLOSE was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2040 Sft_server_close_dir_failed
Level: warning
Origin: Tectia Secure File Transfer

Directory CLOSE failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2041 Sft_server_create_dir
Level: notice
Origin: Tectia Secure File Transfer

Directory CREATE was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2042 Sft_server_create_dir_failed
Level: warning
Origin: Tectia Secure File Transfer

Directory CREATE failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2043 Sft_server_remove_dir
Level: notice
Origin: Tectia Secure File Transfer

Directory REMOVE was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2044 Sft_server_remove_dir_failed
Level: warning
Origin: Tectia Secure File Transfer

Directory REMOVE failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2045 Sft_server_read_dir
Level: informational
Origin: Tectia Secure File Transfer

Directory READ was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2046 Sft_server_read_dir_failed
Level: warning
Origin: Tectia Secure File Transfer

Directory READ failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2047 Sft_server_stream_read
Level: informational
Origin: Tectia Secure File Transfer

File read stream was initiated.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Path	Path to socket file
Offset	File offset
Length	File transfer limit in bytes
Session-Id	Session ID

2048 Sft_server_stream_read_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM READ failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2049 Sft_server_stream_write
Level: informational
Origin: Tectia Secure File Transfer

File write stream was initiated.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Path	Path to socket file
Offset	File offset
Length	File transfer limit in bytes
Session-Id	Session ID

2050 Sft_server_stream_write_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM WRITE failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2051 Sft_server_stream_cancel
Level: informational
Origin: Tectia Secure File Transfer

File STREAM CANCEL was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2052 Sft_server_stream_cancel_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM CANCEL failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2053 Sft_server_stream_wait
Level: informational
Origin: Tectia Secure File Transfer

File STREAM WAIT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle

Argument	Description
Text	Audit message
Session-Id	Session ID

2054 Sft_server_stream_wait_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM WAIT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2055 Sft_server_download_begin
Level: notice
Origin: Tectia Secure File Transfer

File download begins.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Limit	File transfer limit in bytes
Session-Id	Session ID

2056 Sft_server_download_end
Level: notice
Origin: Tectia Secure File Transfer

File download ended.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process

Argument	Description
File name	File name
Read	Number of bytes read
Success | Error	Error code
Session-Id	Session ID

2057 Sft_server_upload_begin
Level: notice
Origin: Tectia Secure File Transfer

File upload begins.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Limit	File transfer limit in bytes
Session-Id	Session ID

2058 Sft_server_upload_end
Level: notice
Origin: Tectia Secure File Transfer

File upload ended.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Wrote	Number of bytes written
Success | Error	Error code
Session-Id	Session ID

2059 Sft_server_statistics
Level: notice
Origin: Tectia Secure File Transfer

File statistics.

Default log facility: daemon

Argument	Description
Username	User's login name

Argument	Description
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Read	Number of bytes read
Wrote	Number of bytes written
Text	Audit message
Session-Id	Session ID

2060 Sft_server_operation_not_allowed
Level: notice
Origin: Tectia Secure File Transfer

Extended SFT protocol message was received, but the operation was prohibited.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Request type	Request type
Request ID	Request ID
Ext request type	Extended request type
Text	Error message
Session-Id	Session ID

2061 Sft_server_operation_no_license
Level: notice
Origin: Tectia Secure File Transfer

Extended SFT protocol message was received, but no suitable license was found for the operation.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Request type	Request type
Request ID	Request ID
Ext request type	Extended request type
Text	Error message
Session-Id	Session ID

2062 Sft_server_no_virtual_folder_dir
Level: error
Origin: Tectia Secure File Transfer

Virtual folder directory does not exist.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
Text	Virtual folder
Directory name	Directory name
Text	Reason

2065 Sft_server_stat_extended
Level: informational
Origin: Tectia Secure File Transfer

File STREAM STAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2066 Sft_server_stat_extended_failed
Level: informational
Origin: Tectia Secure File Transfer

File STREAM STAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2067 Sft_server_lstat_extended
Level: informational
Origin: Tectia Secure File Transfer

File STREAM LSTAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2068 Sft_server_lstat_extended_failed
Level: informational
Origin: Tectia Secure File Transfer

File STREAM LSTAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2069 Sft_server_setstat_extended
Level: notice
Origin: Tectia Secure File Transfer

File STREAM SETSTAT was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2070 Sft_server_setstat_extended_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM SETSTAT failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
Text	Audit message
Session-Id	Session ID

2071 Sft_server_stream_offset
Level: notice
Origin: Tectia Secure File Transfer

File STREAM OFFSET was successful.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2072 Sft_server_stream_offset_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM OFFSET failed.

Default log facility: daemon

Argument	Description
Username	User's login name
Subsystem	Subsystem type
Pid	Process ID of the user process
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2101 Scp_file_upload
Level: notice
Origin: Tectia Server

An SCP file upload has completed successfully or unsuccessfully.

Default log facility: daemon

Argument	Description
Username	User's login name
Session-Id	Session identifier
Status	Result of file transfer
File	Path to the transferred file
Duration	File transfer duration
Data	Bytes copied during file transfer
Speed	File transfer speed (KiB/s)

2102 Scp_file_download
Level: notice
Origin: Tectia Server

An SCP file download has completed successfully or unsuccessfully.

Default log facility: daemon

Argument	Description
Username	User's login name
Session-Id	Session identifier
Status	Result of file transfer
File	Path to the transferred file
Duration	File transfer duration
Data	Bytes copied during file transfer
Speed	File transfer speed (KiB/s)

Appendix E Tectia Mapper Protocol

Tectia Mapper Protocol defines textual communication between Tectia Server and an external application. Requests are sent by Tectia Server via standard output and responses received via Tectia Server's standard input. The external application may be written in any programming language suitable for the task.

The protocol is used in matching local tunnel constraints with external data. (For more information, see **tunnel-local** or the section called "Local Tunnels" in **Tectia Server Configuration** tool.)

E.1 Parameters

Tectia Server and an external application communicate by sending messages of form `<parameter>:<data>`.

The following parameters are valid in the communication (Tectia Server ignores all other parameters):

`version:` *`number`*

Specifies the highest protocol version number understood by the sender. Currently, the only accepted value is "`1`".

`request:` *`id`*

Specifies the request id. The *`id`* is unique for any ongoing requests the external application has not yet completed.

`end-of-request:` *`id`*

Denotes the end of a request.

`success:` *`[additional_info]`*

Denotes a positive response to a request. Including additional information in the data field is optional.

`failure: [additional_info]`

Denotes a negative response to a request. Including additional information in the data field is optional.

Tectia Mapper Protocol is line-based; all messages between Tectia Server and an external application are terminated with a linefeed character (`"\n"` in ANSI C, `0x0A` in hex).

E.2 Communication Between Tectia Server and the External Application

The communication between Tectia Server and an external application (from now on referred to as "application") proceeds as follows (also depicted in Figure E.1):

1. Tectia Server waits for the parameter `version` from the application. The parameter indicates the highest protocol number the application understands. Currently, the only supported version is 1.

2. Tectia Server sends `version`, indicating the highest protocol number Tectia Server understands. The application may ignore this.

3. Tectia Server sends one or more requests. Each request is started by sending `request:id`, where `id` is the request identifier. It is unique for any ongoing requests the application has not yet completed. (The application later sends the request ID back to Tectia Server, see step 4).

   Next, Tectia Server sends zero or more data entries of the form `<key>=<data>`. For a list of the data entries sent by Tectia Server when matching local tunnel constraints with external data, see **tunnel-local** or the section called "Local Tunnels" in **Tectia Server Configuration** tool.)

   The request is terminated by `end-of-request:id` where `id` is the same as in the start of the request.

4. After sending the request(s), Tectia Server waits for the response(s) from the application. If Tectia Server has sent more than one request, the responses may come in any order. Each response starts with `request:id` and is followed by zero or more data entries of the form `<key>=<data>`, containing information the application needs to send to Tectia Server.

   The application ends each response with the parameter `"success:"` or `"failure:"`. This parameter may contain additional information in the data field, for example `"success: Access was allowed"`.

5. Once Tectia Server has received all the responses from the application, it waits for the application to exit. Tectia Server collects the application's exit status and reports an error if the status is not 0.

Note

If the application hangs, Tectia Server will not kill it.

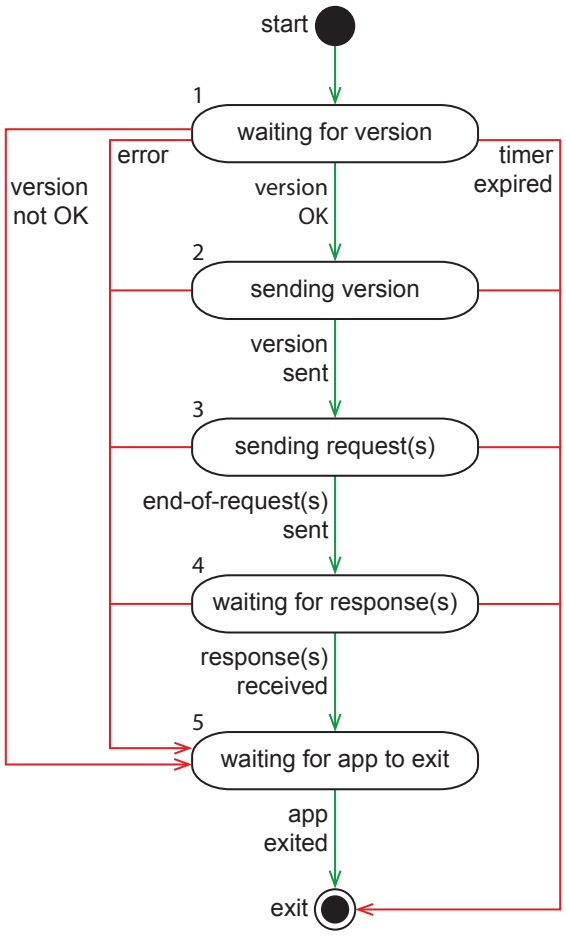


Figure E.1. Tectia Mapper protocol from Tectia Server's perspective

E.3 Examples

This section contains examples of communication sequences between Tectia Server and an external application.

E.3.1 Positive Response

The following figure shows an example of the sequence of messages exchanged between Tectia Server and an external application, resulting in a positive response.

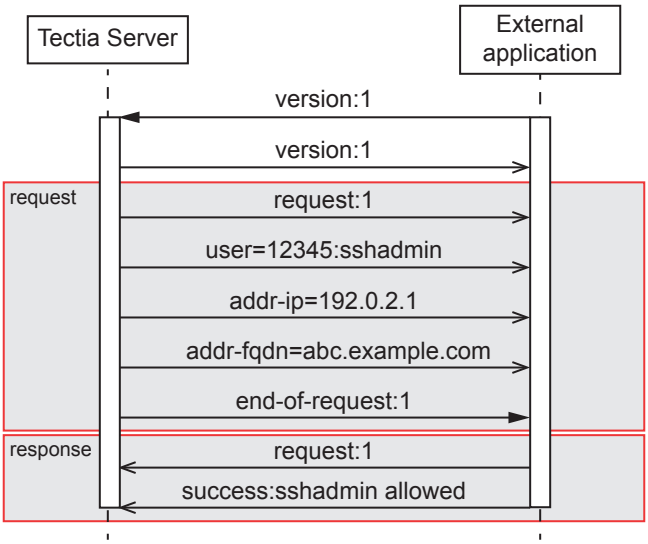


Figure E.2. Sequence diagram of the communication between Tectia Server and an external application

E.3.2 Negative Response

In this example Tectia Server (S) uses an external application (C) to check if user with the id `45678` and user name `jbrown` from the IP address `192.0.2.2` in domain `abc.example.com` is allowed access. The application gives a negative response:

```
C: version:1
S: version:1
S: request:1
S: user=45678:jbrown
S: addr-ip=192.0.2.2
S: addr-fqdn=abc.example.com
S: end-of-request:1
C: request:1
C: failure:jbrown not allowed from that subnet.
```

E.3.3 Checking the Number of Connections

Tectia Server (S) uses an application (C) to check how many connections user `admin` from the IP address `192.0.2.3` has made. The application returns the number of connections:

```
C: version:1
S: version:1
S: request:1
S: user=34567:admin
S: addr-ip=192.0.2.3
S: end-of-request:1
C: request:1
C: number-of-connections=42
```

```
C: success:
```

E.4 Example Application

The following is an example of an external application implemented in Python. You can find it also in `/etc/ssh2/samples/mapper_simple_example.py` on Unix and `<INSTALLDIR>\SSH Tectia AUX \samples\mapper_simple_example.py` on Windows.

```
#!/usr/bin/env python

import sys

# First ssh-server-g3 expects a protocol version number.
# Currently only version 1 is supported.
sys.stdout.write("version:1\n")
sys.stdout.flush()

# Read the version string sent by the server.
version = sys.stdin.readline()
(ver, num) = version.split(':')
if (ver == "version"):
    if (int(num) != 1):
        sys.exit(1)
else:
    sys.exit(1)

# Version is OK, let's wait for the request.
request = sys.stdin.readline()
(request_str, num) = request.split(':')
if (request_str == "request") :
    request_no = int(num)
else:
    sys.exit(2)

# Request started, let's read the request's data.
while 1:
    end_of_request = sys.stdin.readline()
    if not end_of_request:
        break
    if end_of_request.find("end-of-request:",0,15) == 0 :
        (end_of_request_str, num) = end_of_request.split(':')
        end_of_request_no = int(num)
        if (end_of_request_no == request_no) :
            break
        else:
            sys.exit(3)
    else:
        pass # handle the request data

# Request finished, let's send the response.
sys.stdout.write("%s" % request)
sys.stdout.write("success: Well done!\n")
```

```
sys.stdout.flush()

sys.exit(0)
```

Appendix F Removing OpenSSL from Tectia Server

F.1 Background Information

F.1.1 Should I Remove the OpenSSL Library?

When Tectia Server is not used in the FIPS compliant mode (for more information, see **Cryptographic library** and **crypto-lib**), the OpenSSL cryptographic library is not needed and can be removed.

If you do not use the FIPS mode and want to remove OpenSSL from your Tectia Server installation, the following sections provide per-platform instructions for doing it.

F.1.2 What Happens If I Remove the OpenSSL Library?

Once the OpenSSL cryptographic library is removed, if Tectia Server is configured to run in the FIPS compliant mode, it will refuse to start.

F.2 Removing the OpenSSL Cryptographic Library

F.2.1 Linux and Solaris

To remove the OpenSSL cryptographic library from Tectia Server on Unix, first disable FIPS mode, if it has been enabled in configuration.

Remove OpenSSL FIPS libraries with the following commands:

```
# /opt/tectia/sbin/ssh-modeset fips-mode off
```

```
# /opt/tectia/sbin/ssh-modeset fips-remove
```

F.2.2 Windows

To remove the OpenSSL cryptographic library from Tectia Server on Windows, first disable FIPS mode, if it has been enabled, using the Configuration GUI.

Note

If both Tectia Client and Tectia Server are installed, ensure that user-specific Connection Broker configuration(s) have FIPS mode disabled and that the system wide Tectia FIPSMODE switch file is removed. The FIPSMODE file is automatically removed when FIPS mode is disabled from the Tectia Server Configuration GUI (for more information, see **Cryptographic library** and **crypto-lib**).

Changing Optional Installation Modules for Tectia on Windows

To modify Tectia Client and Server optional FIPS module in Windows environment, follow the instructions below:

1. From the Windows **Start** menu, open the **Control Panel** and click **Programs and Features**.
2. In the list of installed programs, select **Tectia Server** and click **Change**.
3. In the installer click **Modify**.
4. Select **Tectia Server** > **FIPS** optional module and change it to *Entire feature will be unavailable*.

Note

If you have installed Tectia Client together with Tectia Server, change also **Tectia Client** > **FIPS** optional module to *Entire feature will be unavailable*.

5. Click **Next** and **Install** to proceed with the *Modify installation* that will remove the Tectia FIPS support module(s).

OpenSSL files removed from Tectia Server on Windows, when FIPS support module is uninstalled:

Note that `<INSTALLDIR>` indicates the default Tectia installation directory on 64-bit Windows versions: `C:\Program Files (x86)\SSH Communications Security\SSH Tectia`

- `<INSTALLDIR>\SSH Tectia AUX\Plugins\<x>.<y>.<z>.<b>\sshcrypto1.dll`

  (*<x>*, *<y>*, *<z>* and *<b>* indicate the Tectia Server version and build numbers, for example `6.6.5.123`.)

- `<INSTALLDIR>\SSH Tectia AUX\fips\fips.dll`
- `<INSTALLDIR>\SSH Tectia AUX\fips\openssl.cnf`

- `<INSTALLDIR>\SSH Tectia AUX\Support binaries\libcrypto-3.dll`
- `<INSTALLDIR>\SSH Tectia AUX\libcrypto-3.dll`
- `<INSTALLDIR>\SSH Tectia Broker\libcrypto-3.dll`
- `<INSTALLDIR>\SSH Tectia Client\libcrypto-3.dll`
- `<INSTALLDIR>\SSH Tectia Server\libcrypto-3.dll`

Appendix G Default and Supported SSH Algorithms

This section describes the SSH algorithms supported by Tectia products.

G.1 Ciphers

Table G.1. Default ciphers (in order of client-side preference)

Name in XML	Name in GUI	FIPS
crypticore128@ssh.com	CryptiCore (Tectia)	
aes128-gcm@openssh.com	AES-128-GCM (OpenSSH)	•
aes256-gcm@openssh.com	AES-256-GCM (OpenSSH)	•
AEAD_AES_128_GCM	AEAD_AES_128_GCM	•
AEAD_AES_256_GCM	AEAD_AES_256_GCM	•
aes128-ctr	AES-128-CTR	•
aes192-ctr	AES-192-CTR	•
aes256-ctr	AES-256-CTR	•

Table G.2. All supported ciphers

Name in XML	Name in GUI	FIPS
3des-cbc	3DES	•
AEAD_AES_128_GCM	AEAD_AES_128_GCM	•
AEAD_AES_256_GCM	AEAD_AES_256_GCM	•
aes128-cbc	AES-128-CBC	•
aes128-ctr	AES-128-CTR	•

Name in XML	Name in GUI	FIPS
aes128-gcm@openssh.com	AES-128-GCM (OpenSSH)	•
aes192-cbc	AES-192-CBC	•
aes192-ctr	AES-192-CTR	•
aes256-cbc	AES-256-CBC	•
aes256-ctr	AES-256-CTR	•
aes256-gcm@openssh.com	AES-256-GCM (OpenSSH)	•
arcfour	Arcfour	
blowfish-cbc	Blowfish	
crypticore128@ssh.com	CryptiCore (Tectia)	
seed-cbc@ssh.com	SEED (Tectia)	
twofish128-cbc	Twofish-128	
twofish192-cbc	Twofish-192	
twofish256-cbc	Twofish-256	
twofish-cbc	Twofish	

G.2 Key-Exchange Algorithms

Table G.3. Default KEXs (in order of client-side preference)

Name in XML	Name in GUI	FIPS
mlkem1024nistp384-sha384	PQC: mlkem1024nistp384-sha384	•
mlkem768nistp256-sha256	PQC: mlkem768nistp256-sha256	•
mlkem768x25519-sha256	PQC: mlkem768x25519-sha256	•
ecdh-nistp521-kyber1024-sha512@ssh.com	PQC: ecdh-nistp521-kyber1024-sha512 (Tectia)	•
curve25519-frodokem1344-sha512@ssh.com	PQC: curve25519-frodokem1344-sha512 (Tectia)	•
sntrup761x25519-sha512@openssh.com	PQC: sntrup761x25519-sha512 (OpenSSH)	•
diffie-hellman-group-exchange-sha256	DH-GEX-SHA256	
diffie-hellman-group16-sha512	DH-Group16-SHA512	•
diffie-hellman-group18-sha512	DH-Group18-SHA512	•
diffie-hellman-group14-sha256	DH-Group14-SHA256	•
diffie-hellman-group14-sha256@ssh.com	DH-Group14-SHA256 (Tectia)	•
curve25519-sha256	Curve25519-sha256	•
curve25519-sha256@libssh.org	Curve25519-sha256 (libssh)	•

Table G.4. All supported KEXs

Name in XML	Name in GUI	FIPS
curve25519-frodokem1344-sha512@ssh.com	PQC: curve25519-frodokem1344-sha512 (Tectia)	•

Name in XML	Name in GUI	FIPS
curve25519-sha256	Curve25519-sha256	•
curve25519-sha256@libssh.org	Curve25519-sha256 (libssh)	•
curve448-kyber1024-sha512@ssh.com	PQC: curve448-kyber1024-sha512 (Tectia)	•
diffie-hellman-group14-sha1	DH-Group14-SHA1	•
diffie-hellman-group14-sha224@ssh.com	DH-Group14-SHA224 (Tectia)	•
diffie-hellman-group14-sha256	DH-Group14-SHA256	•
diffie-hellman-group14-sha256@ssh.com	DH-Group14-SHA256 (Tectia)	•
diffie-hellman-group15-sha256@ssh.com	DH-Group15-SHA256 (Tectia)	•
diffie-hellman-group15-sha384@ssh.com	DH-Group15-SHA384 (Tectia)	•
diffie-hellman-group16-sha384@ssh.com	DH-Group16-SHA384 (Tectia)	•
diffie-hellman-group16-sha512	DH-Group16-SHA512	•
diffie-hellman-group16-sha512@ssh.com	DH-Group16-SHA512 (Tectia)	•
diffie-hellman-group18-sha512	DH-Group18-SHA512	•
diffie-hellman-group18-sha512@ssh.com	DH-Group18-SHA512 (Tectia)	•
diffie-hellman-group1-sha1	DH-Group1-SHA1	
diffie-hellman-group-exchange-sha1	DH-GEX-SHA1	
diffie-hellman-group-exchange-sha224@ssh.com	DH-GEX-SHA224 (Tectia)	
diffie-hellman-group-exchange-sha256	DH-GEX-SHA256	
diffie-hellman-group-exchange-sha384@ssh.com	DH-GEX-SHA384 (Tectia)	
diffie-hellman-group-exchange-sha512@ssh.com	DH-GEX-SHA512 (Tectia)	
ecdh-nistp521-kyber1024-sha512@ssh.com	PQC: ecdh-nistp521-kyber1024-sha512 (Tectia)	•
ecdh-sha2-nistp256	ECDH-NISTP256	•
ecdh-sha2-nistp384	ECDH-NISTP384	•
ecdh-sha2-nistp521	ECDH-NISTP521	•
mlkem1024nistp384-sha384	PQC: mlkem1024nistp384-sha384	•
mlkem768nistp256-sha256	PQC: mlkem768nistp256-sha256	•
mlkem768x25519-sha256	PQC: mlkem768x25519-sha256	•
sntrup761x25519-sha512@openssh.com	PQC: sntrup761x25519-sha512 (OpenSSH)	•

G.3 Message-Authentication Codes

Table G.5. Default MACs (in order of client-side preference)

Name in XML	Name in GUI	FIPS
crypticore-mac@ssh.com	CryptiCore (Tectia)	
hmac-sha2-256	HMAC-SHA2-256	•

Name in XML	Name in GUI	FIPS
hmac-sha256-2@ssh.com	HMAC-SHA256-2 (Tectia)	•
hmac-sha2-512	HMAC-SHA2-512	•
hmac-sha512@ssh.com	HMAC-SHA512 (Tectia)	•
hmac-sha1	HMAC-SHA1	•

Table G.6. All supported MACs

Name in XML	Name in GUI	FIPS
crypticore-mac@ssh.com	CryptiCore (Tectia)	
hmac-md5	HMAC-MD5	
hmac-md5-96	HMAC-MD5-96	
hmac-md5-96-etm@openssh.com	HMAC-MD5-96-ETM (OpenSSH)	
hmac-md5-etm@openssh.com	HMAC-MD5-ETM (OpenSSH)	
hmac-sha1	HMAC-SHA1	•
hmac-sha1-96	HMAC-SHA1-96	
hmac-sha1-96-etm@openssh.com	HMAC-SHA1-96-ETM (OpenSSH)	
hmac-sha1-etm@openssh.com	HMAC-SHA1-ETM (OpenSSH)	•
hmac-sha224@ssh.com	HMAC-SHA224 (Tectia)	
hmac-sha2-256	HMAC-SHA2-256	•
hmac-sha2-256-etm@openssh.com	HMAC-SHA2-256-ETM (OpenSSH)	•
hmac-sha2-512	HMAC-SHA2-512	•
hmac-sha2-512-etm@openssh.com	HMAC-SHA2-512-ETM (OpenSSH)	•
hmac-sha256@ssh.com	HMAC-SHA256 (Tectia/Old)	
hmac-sha256-2@ssh.com	HMAC-SHA256-2 (Tectia)	
hmac-sha384@ssh.com	HMAC-SHA384 (Tectia)	
hmac-sha512@ssh.com	HMAC-SHA512 (Tectia)	

G.4 Host-Key and Public Key Signature Algorithms

Table G.7. Default host-key algorithms (in order of client-side preference)

Name in XML	Name in GUI	FIPS
rsa-sha2-512	rsa-sha2-512	•
rsa-sha2-256	rsa-sha2-256	•
ssh-rsa-sha256@ssh.com	ssh-rsa-sha256 (Tectia)	•
ecdsa-sha2-nistp521	ecdsa-sha2-nistp521	•
ecdsa-sha2-nistp384	ecdsa-sha2-nistp384	•
ecdsa-sha2-nistp256	ecdsa-sha2-nistp256	•
x509v3-sign-rsa-sha256@ssh.com	x509v3-sign-rsa-sha256 (Tectia)	•
x509v3-ecdsa-sha2-nistp256	x509v3-ecdsa-sha2-nistp256	•

Name in XML	Name in GUI	FIPS
x509v3-ecdsa-sha2-nistp384	x509v3-ecdsa-sha2-nistp384	•
x509v3-ecdsa-sha2-nistp521	x509v3-ecdsa-sha2-nistp521	•
x509v3-rsa2048-sha256	x509v3-rsa2048-sha256	•
ssh-ed25519	ssh-ed25519	•
ecdsa-sha2-nistp256-cert-v01@openssh.com	ecdsa-sha2-nistp256-cert-v01@openssh.com	•
ecdsa-sha2-nistp384-cert-v01@openssh.com	ecdsa-sha2-nistp384-cert-v01@openssh.com	•
ecdsa-sha2-nistp521-cert-v01@openssh.com	ecdsa-sha2-nistp521-cert-v01@openssh.com	•
ssh-ed25519-cert-v01@openssh.com	ssh-ed25519-cert-v01@openssh.com	•
rsa-sha2-256-cert-v01@openssh.com	rsa-sha2-256-cert-v01@openssh.com	•
rsa-sha2-512-cert-v01@openssh.com	rsa-sha2-512-cert-v01@openssh.com	•

Table G.8. All supported host-key and public key signature algorithms

Name in XML	Name in GUI	FIPS
ecdsa-sha2-nistp256	ecdsa-sha2-nistp256	•
ecdsa-sha2-nistp256-cert-v01@openssh.com	ecdsa-sha2-nistp256-cert-v01@openssh.com	•
ecdsa-sha2-nistp384	ecdsa-sha2-nistp384	•
ecdsa-sha2-nistp384-cert-v01@openssh.com	ecdsa-sha2-nistp384-cert-v01@openssh.com	•
ecdsa-sha2-nistp521	ecdsa-sha2-nistp521	•
ecdsa-sha2-nistp521-cert-v01@openssh.com	ecdsa-sha2-nistp521-cert-v01@openssh.com	•
rsa-sha2-256	rsa-sha2-256	•
rsa-sha2-256-cert-v01@openssh.com	rsa-sha2-256-cert-v01@openssh.com	•
rsa-sha2-512	rsa-sha2-512	•
rsa-sha2-512-cert-v01@openssh.com	rsa-sha2-512-cert-v01@openssh.com	•
ssh-dss	ssh-dss	
ssh-dss-cert-v01@openssh.com	ssh-dss-cert-v01@openssh.com	
ssh-dss-sha224@ssh.com	ssh-dss-sha224 (Tectia)	•
ssh-dss-sha256@ssh.com	ssh-dss-sha256 (Tectia)	•
ssh-dss-sha384@ssh.com	ssh-dss-sha384 (Tectia)	•
ssh-dss-sha512@ssh.com	ssh-dss-sha512 (Tectia)	•
ssh-ed25519	ssh-ed25519	•
ssh-ed25519-cert-v01@openssh.com	ssh-ed25519-cert-v01@openssh.com	•
ssh-rsa	ssh-rsa	
ssh-rsa-cert-v01@openssh.com	ssh-rsa-cert-v01@openssh.com	
ssh-rsa-sha224@ssh.com	ssh-rsa-sha224 (Tectia)	•
ssh-rsa-sha256@ssh.com	ssh-rsa-sha256 (Tectia)	•
ssh-rsa-sha384@ssh.com	ssh-rsa-sha384 (Tectia)	•
ssh-rsa-sha512@ssh.com	ssh-rsa-sha512 (Tectia)	•
x509v3-ecdsa-sha2-nistp256	x509v3-ecdsa-sha2-nistp256	•
x509v3-ecdsa-sha2-nistp384	x509v3-ecdsa-sha2-nistp384	•

Name in XML	Name in GUI	FIPS
x509v3-ecdsa-sha2-nistp521	x509v3-ecdsa-sha2-nistp521	•
x509v3-rsa2048-sha256	x509v3-rsa2048-sha256	•
x509v3-sign-dss	x509v3-sign-dss	
x509v3-sign-dss-sha224@ssh.com	x509v3-sign-dss-sha224 (Tectia)	•
x509v3-sign-dss-sha256@ssh.com	x509v3-sign-dss-sha256 (Tectia)	•
x509v3-sign-dss-sha384@ssh.com	x509v3-sign-dss-sha384 (Tectia)	•
x509v3-sign-dss-sha512@ssh.com	x509v3-sign-dss-sha512 (Tectia)	•
x509v3-sign-rsa	x509v3-sign-rsa	
x509v3-sign-rsa-sha224@ssh.com	x509v3-sign-rsa-sha224 (Tectia)	•
x509v3-sign-rsa-sha256@ssh.com	x509v3-sign-rsa-sha256 (Tectia)	•
x509v3-sign-rsa-sha384@ssh.com	x509v3-sign-rsa-sha384 (Tectia)	•
x509v3-sign-rsa-sha512@ssh.com	x509v3-sign-rsa-sha512 (Tectia)	•
x509v3-ssh-dss	x509v3-ssh-dss	
x509v3-ssh-rsa	x509v3-ssh-rsa	

Appendix H Open Source Software License Acknowledgements

SSH Communications Security Corporation acknowledges the following Open Source Software used in the Tectia client/server solution.

BSD Software

> This product includes software developed by the University of California, Berkeley and its contributors.

DES

> This product includes software developed by Eric Young `eay@cryptsoft.com`.

ICU

> Copyright © 1995-2016 International Business Machines Corporation and others
>
> All rights reserved.
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND

NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

OpenSSL

OpenSSL 3.0 is licensed under the Apache License 2.0.

Apache License Version 2.0, January 2004 https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this

License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such

NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor

harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

PCRE

This software includes PCRE library. Copyright © 1997-2015 University of Cambridge. All rights reserved.

C++ wrapper functions. Copyright © 2007-2015, Google Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

XFree86

This Software contains portions of XFree86 software and the delivery of XFree86 software or portions of the said software is subject to the acknowlegement of the following copyright notice and permission notice of The Open Group:

Copyright © 1988, 1998 The Open Group

Permission to use, copy, modify, distribute, and sell XFree86 software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both the copyright notice and this permission notice appear in supporting documentation.

THE XFREE86 SOFTWARE IS PROVIDE "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE

WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE XFREE86 SOFTWARE OR THE USE OR OTHER DEALINGS IN THE XFREE86 SOFTWARE.

Except as contained in this notice, the name of The Open Group shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from The Open Group.

ZLIB

This software incorporates zlib data compression library by Jean-loup Gailly and Mark Adler.

liboqs

Licensed under MIT. Copyright (c) 2016-2024 Open Quantum Safe project.

liboqs includes some third party libraries or modules that are licensed differently, including:

- BSD 3-Clause License
- Apache License v2.0
- public domain
- BSD-like CRYPTOGAMS license
- CC0
- Custom license for rand_nist.c:

  Created by Bassham, Lawrence E (Fed) on 8/29/17.

  Copyright © 2017 Bassham, Lawrence E (Fed). All rights reserved.

  NIST-developed software is provided by NIST as a public service. You may use, copy, and distribute copies of the software in any medium, provided that you keep intact this entire notice. You may improve, modify, and create derivative works of the software or any portion of the software, and you may copy and distribute such modifications or works. Modified works should carry a notice stating that you changed the software and should note the date and nature of any such change. Please explicitly acknowledge the National Institute of Standards and Technology as the source of the software.

  NIST-developed software is expressly provided "AS IS." NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED, IN FACT, OR ARISING BY OPERATION OF LAW, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, AND DATA ACCURACY. NIST NEITHER REPRESENTS NOR WARRANTS THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE,

OR THAT ANY DEFECTS WILL BE CORRECTED. NIST DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF THE SOFTWARE OR THE RESULTS THEREOF, INCLUDING BUT NOT LIMITED TO THE CORRECTNESS, ACCURACY, RELIABILITY, OR USEFULNESS OF THE SOFTWARE.

You are solely responsible for determining the appropriateness of using and distributing the software and you assume all risks associated with its use, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and the unavailability or interruption of operation. This software is not intended to be used in any situation where a failure could cause risk of injury or damage to property. The software developed by NIST employees is not subject to copyright protection within the United States.

SPDX-License-Identifier: Unknown

Modified for liboqs by Douglas Stebila

Appendix I Changing the Host Key of Tectia Server

Tectia Server version 6.4.19 has a feature to help with changing the host keys on the client side. To use it, you can configure host key rotation on the server-side. This will allow clients that authenticate the server with the old host key to save the new host key after successful user authentication and delete the old one once the old key is removed on the server-side, for example after 3 months. This feature requires a Tectia client version 6.4.19 that has Host Key Policy Rotation enabled (by default enabled when connecting to Tectia servers only) or a OpenSSH client version 6.8 or above that has UpdateHostKeys enabled.

Quick Comparison:

Manual Change

- Change host key without advertising it first. All secure shell clients that have previously connected and saved the old host key to known hosts fail to connect or prompt a host key changed warning.

Automatic Rotation

- Time-based key generation, advertising and rotation that changes the host key
- hostkey (current advertised and used as server identity)
- hostkey.next (new advertised)
- hostkey.old (previous hostkey that has been removed from configuration)
- Same algorithm and key size as the current hostkey
- Must not be enabled for Tectia Server cluster nodes

- Server_hostkey_rotation_started and Server_hostkey_rotation audit messages

Manual Rotation

- Administrator controls new key generation, advertising and changing the host key
- Host key algorithm or key size can be different for new key
- If Tectia Server is part of a cluster the new host key has to be shared on all nodes and advertising needs to be enabled and disabled for all keys on a node. Advertising must not be enabled on other node unless it has the same current and new host keys. Also, advertising must be disabled for all keys on a node before new key is taken into active use and advertising can only be enabled again once all nodes have taken the same new key into active use.

I.1 Host key Algorithm in Manual Host Key Rotation

The hostkey algorithm should be decided based on security policy. The new host key with different algorithm could be taken into active use faster in the environment than using the same algorithm as the current host key but it may cause unexpected key exchange failures if the different algorithms are not allowed in configuration.

While Tectia Server can have multiple identities, the client and server can only agree on one hostkey algorithm during key exchange in secure shell protocol. The negotiated algorithm depends on what the server offers and what the client supports and prefers in its configuration and what type host key(s), if any, it has saved to known hosts for the server.

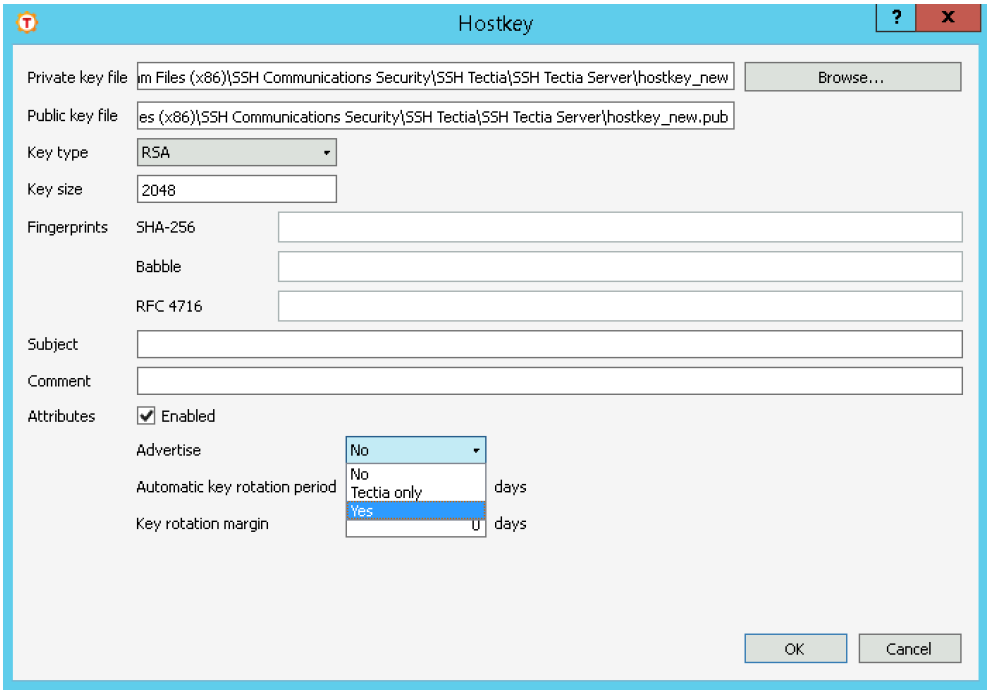
For example if Tectia Server has currently RSA hostkey and new host key is generated with different algorithm for example ECDSA or ED25519 both can be enabled simultaneously as current hostkey identities. Clients that connect the first time may already use the new hostkey but clients that prefer or only support RSA or clients that have connected before continue to use the RSA hostkey as long as the server has it enabled and offers it in key exchange.

Tectia Server can be configured so that current RSA hostkey is enabled but not advertised and the new hostkey of different algorithm is both enabled and advertised. This allows secure shell clients that have connected before and support and enable Host Key Rotation / UpdateHostKeys to connect once and authenticate the server with RSA hostkey and after successful user authentication to add the advertised hostkey and remove the old RSA hostkey from known hosts within the same connection. For subsequent connections by this client the new hostkey is used provided that the client allows it in configuration.

If the same algorithm is used for the new host key, for example current hostkey is RSA and new RSA hostkey is generated, then only the first one is enabled as hostkey identity. In this case Tectia Server must be configured so that current RSA hostkey is enabled and advertised and the new RSA hostkey is advertised. This ensures secure shell clients that support and enable Host Key Rotation / UpdateHostKeys can connect and authenticate the server with current RSA hostkey and after successful user authentication add the advertised hostkey for future use when the old RSA hostkey is removed from Tectia Server.

I.2 Manual Rotation Example using RSA Host Keys

1. After upgrade create a new RSA host key **hostkey_new** in Tectia Server Configuration GUI → Identity → Generate key



   Or on command-line:

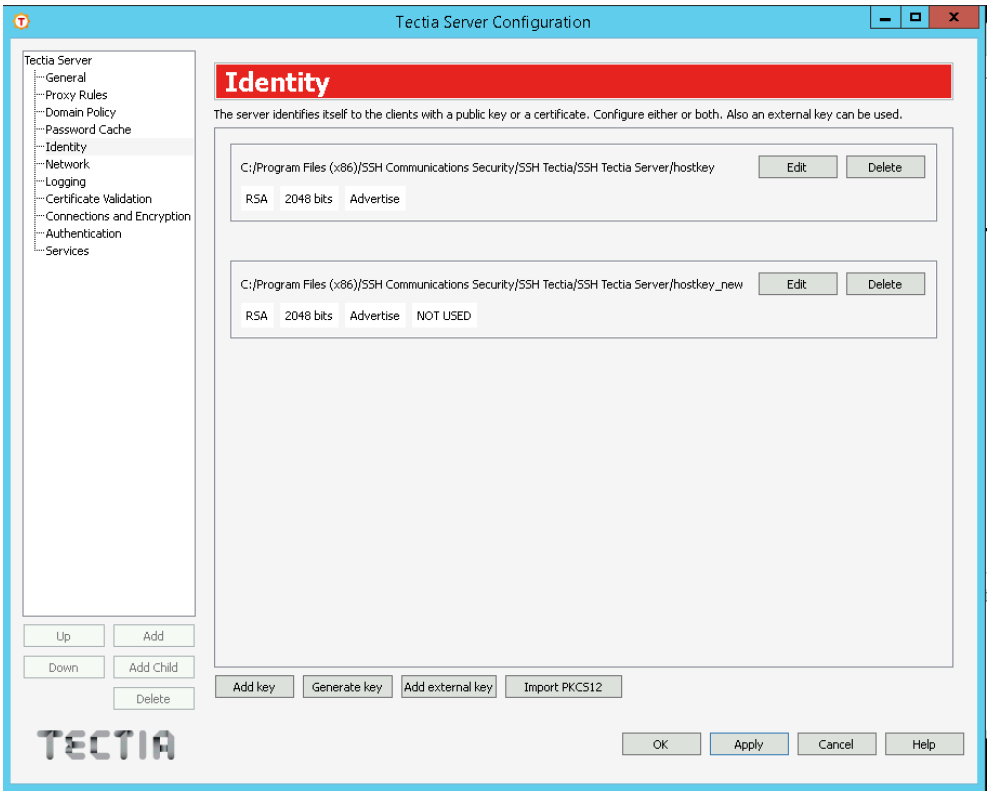
```
ssh-keygen-g3 -H -P hostkey_new
```

2. Advertise Current and New Host Key During Renewal Period Before Rotation

   In Tectia Server Configuration GUI → Identity → Edit **hostkey** and **hostkey_new**

   Change Advertise to **Yes** for both current host key and new host key so that they are advertised to secure shell clients and **Apply** to reload the configuration.

   The current host key is used in server authentication and new host key is saved on the client-side for future use by the clients that support this. Note that if current host key has Advertise set to **No** (default), then clients will remove it prematurely and the next connection will result in save host key prompt.



Or edit *ssh-server-config.xml*

```
<hostkey advertise="yes">
   <private file="C:\Program Files (x86)\SSH Communications Security\
                  SSH Tectia\SSH Tectia Server\hostkey" />
   <public file="C:\Program Files (x86)\SSH Communications Security\
                 SSH Tectia\SSH Tectia Server\hostkey.pub" />
</hostkey>
```

To reconfigure Tectia Server on command-line:

```
ssh-server-ctl reload
```

3. Monitor Tectia Server Logs

   Tectia Server logs Hostkey-advert-accepted informational audit message when a client saves advertised new host key and server has proved ownership. These messages can be used to track the adoption of the new host key in the environment over time.

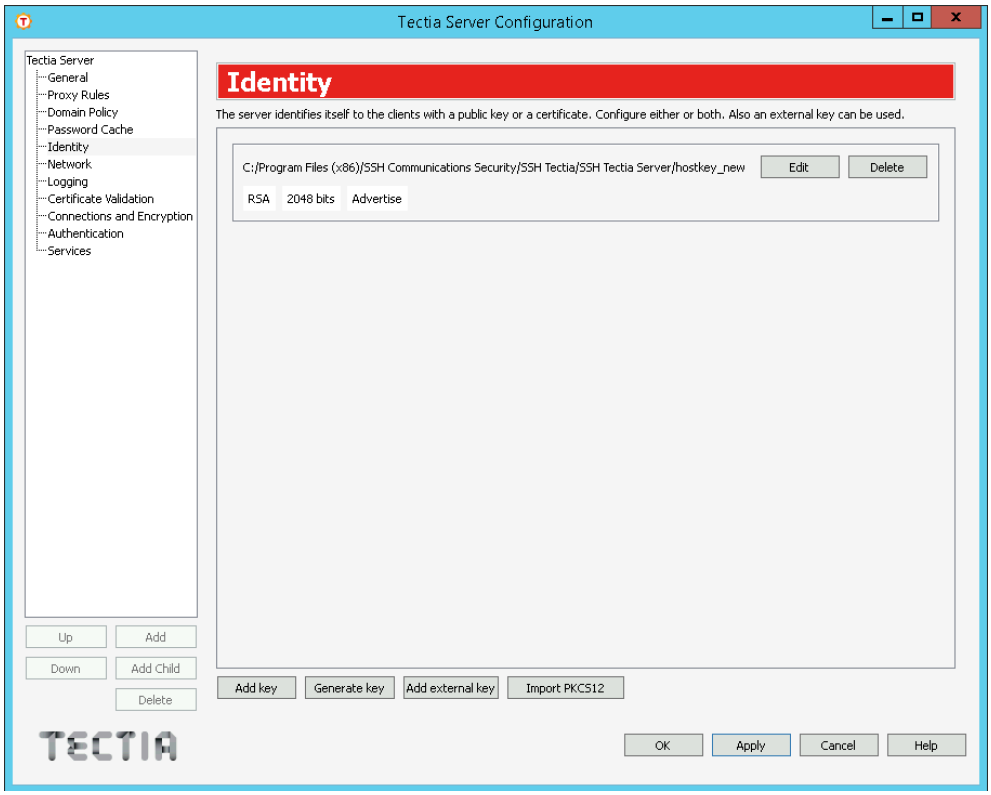
```
132 Hostkey_advert_accepted, Username: <authenticated_user>,
Src IP: 127.0.0.1, Src Port: 50737, "xicaz-...-saxux",
"SHA-256: gThQLwq2eiVIRd3T8X4JElNr9SGa7WRcUX93Audbe2I", Session-Id: 147,
Protocol-session-Id: C85232F7554FBB5814ADAAD6D8E0...
```

4. Host Key Rotation

   When Tectia Server's old key is removed from configuration and new key taken into active use, any secure shell clients that have not connected during the renewal period or clients that do not support or

do not have Host Key Rotation / UpdateHostKeys enabled and have previously connected and saved the old host key to known hosts fail to connect or prompt a host key changed warning.

In Tectia Server Configuration **GUI→Identity→Delete** “hostkey” to remove the old key from configuration and **Apply** to reload the configuration so that the **hostkey_new** becomes the current host key.



Or edit *ssh-server-config.xml* so that only **hostkey_new** key pair is specified:

```
                  <hostkey advertise="yes">
   <private file="C:\Program Files (x86)\SSH Communications Security\
SSH Tectia\SSH Tectia Server\hostkey_new" />
   <public file="C:\Program Files (x86)\SSH Communications Security\
SSH Tectia\SSH Tectia Server\hostkey_new.pub" />
 </hostkey>
```

To reconfigure Tectia Server on command-line:

```
ssh-server-ctl reload
```

Rename **hostkey** to **hostkey_removed** and **hostkey.pub** to **hostkey_removed.pub** in installation directory so that it is not accidentally taken into use if Tectia Server is started without configuration file. Note also that next Tectia Server upgrade will automatically generate a new **hostkey** if the file with this name does not already exist.

If the current **hostkey_new** is advertised, then after successful user authentication clients that support and allow this in configuration will automatically remove from known hosts any host keys that are no longer advertised.

Tectia Client 6.4.19 has Host Key Policy Rotation enabled by default when connecting to Tectia servers only and Tectia Client attempts to remove the old keys from all known host key locations. On the client-side the following command can be used to view keys in local host key store(s):

```
ssh-keygen-g3 -F host_id
```

where *host_id* is **hostname** or **address#port**

I.3 Fingerprints

The administrator can notify the users via some unalterable method of the expected fingerprint of the new host key and information when the new key will be taken into use.

The displayed fingerprint type on the client-side depends on the implementation and version of the client. For example recent Tectia Clients by default show both the Babble format and SHA256 base64 format fingerprints, recent OpenSSH clients show SHA256 base64 format fingerprint and PuTTY shows the RFC 4716 format fingerprint.

To obtain the fingerprints in Tectia Server Configuration GUI → Identity > Edit shows all three fingerprints of the current host key and any other host keys that are explicitly configured. Alternatively, on the Tectia Server command-line:

```
ssh-keygen-g3 --hash sha256 --fingerprint-type base64 -F hostkey_new.pub
ssh-keygen-g3 -F hostkey_new.pub
ssh-keygen-g3 --rfc4716 -F hostkey_new.pub
ssh-keygen-g3 --hash sha1 --fingerprint-type hex -F hostkey_new.pub
```

or

```
ssh-server-ctl status
```

Output shows SHA256 fingerprints for configured and any .next host keys used in automated rotation if enabled.

I.4 Replacing Host Public Key on Client-Side

For file transfer scripts or other non-interactive users, the public host key needs to be replaced on the client-side for clients that do not support hostkey rotation, for example file transfer jobs originating from Tectia SSH Server on IBM z/OS.

After the host key change client-side tools that obtain the current host key from the server like Tectia ssh-broker-ctl probe-key or ssh-keyfetch can be used. The following command can be used to view keys in local host key store(s) for the server:

```
ssh-keygen-g3 -F host_id
```

where *host_id* is hostname or address#port, e.g. serverhost

I.4.1 z/OS Example

Verify the fingerprint automatically and replace the key, for example z/OS Tectia SSH Server version 6.6.9:

```
ssh-broker-ctl probe-key --hostkey-fp=expected-fingerprint \
--save-hostkey serverhost
```

The ssh-keyfetch tool can be used with Tectia SSH Server version 6.6.8 and below on IBM z/OS.

I.4.2 Windows Tectia Client Example

Replace the key hashed format and verify the fingerprint manually from output:

```
ssh-keyfetch --append=no -a -f hashed serverhost
```

Index

Symbols

$HOME, 11
%APPDATA%, 11
%USERPROFILE%, 11
<INSTALLDIR>, 11

A

access rules, 69, 104
Active Directory, 74, 75
address, listen, 64, 121, 220
address family, 110
administrators, 219
advanced GUI mode, 47
agent forwarding, 158, 247
AIX
  installation, 22
  uninstallation, 33
AIX LAM, 146
allowed hosts, 220
allowing commands, 95, 157
allowing subsystems, 102, 154
allowing terminal access, 91, 153
allowing tunneling
  local tunnels, 96, 158
  remote tunnels, 99, 161
APPDATA, 11
application tunneling, 235
auditing, 223
  logins, 226
audit message reference, 341
audit messages, SFTP, 93, 154
authentication, 80, 167
  certificate, 171, 178, 178
  GSSAPI, 201
  host-based, 188
  host-based with certificates, 190
  Kerberos, 201, 206
  keyboard-interactive, 193, 194, 197, 199, 200
  LAM, 193, 200
  PAM, 145, 193, 193, 194
  password, 84, 88, 144, 145, 173, 175, 193
  public-key, 84
  server, 60, 118, 168, 175
  user, 142, 247
  RADIUS, 88, 146, 193, 199
  SecurID, 88, 145, 193, 197
  using external application, 202
authentication chain, 134, 203
authentication forwarding, 206, 247
authentication methods, 84, 133, 167
authority info access, 172, 178
authorization file, 84, 143, 175, 299, 300
authorized_keys directory, 84, 143, 175, 299
authorized_keys file, 176, 300
authorized keys on network drive, 177
automated file transfer, 234
auxiliary data directory
  on Unix, 36, 109
  on Windows, 38

B

banner message, 52, 133, 224
basic configuration, 45
blackboard, adding to, 142
BSM (Solaris Auditing), 226

C

CA certificate, 68, 127, 178
  intermediate, , 127
  trusted, , 127
certificate authentication
  server, 60, 119, 171
  user, 65, 124, 178, 178, 181
certificate cache file, 67, 126
certificate revocation list (CRL)
  auto update, 67, 127
  disabling, , 127
  distribution point, 172, 178
  prefetching, 68, 127
certificates
  enrolling, 172
  revoked, 172
  validating, 65, 124
certificates in host-based authentication, 190
certificate viewer, 336
certification, FIPS 140-2, 51, 110
certification authority (CA), 65, 124, 171

changing expired passwords, 163
changing host key, 170
channel, 235
chroot, 176
chrooting, 220, 222
ciphers, 80, 80, 131
clients
  CMP enrollment, 325
CMP enrollment client, 325
command-line tools, 293
commands, 95, 157
configuration
  in multiple files, 103
  server, 45
configuration file
  backing up, 18
  divided, 103
  permissions, 37, 258
  server, 35, 102, 109
  syntax, 275
configuration tool, 45
configuring
  RSA Agent, 199
  selectors, 104
  server, 45
connection rules, 76, 130
connections
  maximum number, 50, 123
  total number (per servant), 50, 124
CRL (certificate revocation list)
  auto update, 67, 127
  disabling, , 127
  distribution point, 172, 178
  prefetching, 68, 127
CryptiCore, 228
cryptographic library, 51, 110
customer support, 11

D

Debian, 27
  server installation, 27
DEB packages, 27
debugging
  on Unix, 250
  on Windows, 252
  running service, 249, 251
  user authentication with certificates, 187
debug log, 47
default-path, 115
default port, 64, 121
default settings, restoring, 47
denial-of-service attack, 50, 129
denying commands, 95, 157, 230
denying connection attempts, 220
denying file transfers, 237
denying subsystems, 102, 154
denying terminal access, 91, 153, 229, 237
denying tunneling, 229
  local tunnels, 96, 158
  remote tunnels, 99, 161
detecting dead connections, 79
Diffie-Hellman key exchange, 171
digital signature in key usage, 127
directories
  installation (Unix), 36
  installation (Windows), 37
  profile, 176
  root, 176
  virtual, 94, 230
disabling CRL, 69, 127
discard limit, 50, 129
disclaimer before login, 224
disk space requirement, 16
documentation, 9
documentation conventions, 10
Document Type Definition (DTD), 275
DoD PKI, 67, 127
domain access with one-way trust, 57, 122
domain controller, 175
domain policy, 55, 121
domain user account, 57, 122, 175, 176
downloading software, 21
DSA key
  private,
  public,
DSA key pair, 169

E

ECDSA key
  private,

public,
ECDSA key pair, 169
Ed25519 key
private,
public,
editing selectors, 69
empty passwords, 174
encryption, 80
Enforce digital signature in key usage, 67
enrolling host certificate, 172
environment variables, 11, 101, 153
ssh-keyfetch, 323
ssh-server-g3, 297
error situations, 257
event log, 31, 47, 65, 122, 223, 341
expired CRL, , 127
expired passwords, 163, 174
external host key, 173
PKCS, 119
external key viewer, 340

F

failed login, 207
failure
last login time, 259
password authentication, 258
server not starting, 257, 258
virtual folders, 259
Federal Information Processing Standard (FIPS), 51, 110
file locations
on Unix, 36
on Windows, 37
file permissions, 35
file transfer, 227
automated, 234
fingerprint, 170, 316, 316, 318
FIPS 140-2, 51, 110
FIPS 140-2 mode
enabling, 51
FIPS mode
enabling, 110
firewall, 179
folders
virtual, 94, 230
forced commands, 95, 157, 222
forwarding, 235
agent, 158, 247
authentication, 247
local, 238
remote, 243
X11, 158, 245
FreeRADIUS, 199
fully qualified domain name (FQDN), 73, 98, 100, 172, 189, 299

G

generating host key, 169
Generic Security Service API (GSSAPI), 201
getting started with Tectia Server, 41
group, 83, 88, 135, 152, 175
GSSAPI authentication, 86, 147, 201
GUI mode, 47

H

hardware requirement, 16
Hexl, 337
HOME, 11
home folder, 94, 230
host-based authentication, 86, 144, 188, 299
host-based authentication in FIPS mode, 188
host-based authentication with certificates, 190
host certificate, 119
enrolling, 172
host key, 35, 60, 117
changing, 170
external, 119, 173
generating, 169
multiple, 169
private, 118
public, 118
host key algorithms, 80, 80, 132
HP-UX
installation, 24
uninstallation, 33
HTTP proxy URL, 66, 124
HTTP repository, 172, 178

I

IAS, 199

IBM AIX, 22
identity, 60
ignore-nisplus-no-permission, 115
ignoring AIX login restriction, 114
ignoring AIX rlogin restriction, 113
incoming tunnels, 243
installation
  optional module, 406
  planning, 15
  silent, 32
  upgrading, 19
installation packages, 17
INSTALLDIR, 11
installed files, 35
installing
  server on Debian, 27
installing Tectia Server
  on AIX, 22
  on HP-UX, 24
  on Linux, 25
  on Solaris, 28
  on Windows, 30
interactive forced commands, 95, 157
intermediate CA certificate, , 127
Internet Authentication Service (IAS), 199
invalid host key permissions, 257

K

keepalive, 79, 130
Kerberos, 147
Kerberos authentication, 201, 206
KEXs, 80, 80, 132
key
  host, 60, 117
keyboard-interactive authentication, 87, 145, 193, 194, 197, 199, 200
key exchange, 171
key fingerprint, 170, 316, 316, 318
key generation, 169
key rotation, 60, 117, 171, 423

L

LAM authentication, 193, 200
LDAP servers, 67, 126, 179
legal disclaimer, 224
library, cryptographic, 51, 110
license file, 17, 35
licensing, 17
Lightweight Directory Access Protocol (LDAP), 172, 178
Linux
  installation, 25
  uninstallation, 34
listen address, 64, 121, 220
listener, 63, 121
load control, 50, 129
local port forwarding, 238
local tunnels, 96, 158, 235, 238
local user account, 174
location, installed files, 35
logging, 47, 65, 122, 223, 341
  customizing, 224
login failure, 207
login grace time, 52, 133
log message reference, 341
log on locally rights, 175

M

MACs, 80, 80, 131
maintenance release, 21
man-in-the-middle attack, 170, 171
man pages, 293
mapper
  configuration element, 147, 202
  protocol, 399
maximum number of connections, 50, 123
maximum number of new connections, 123
maximum number of processes, 50, 123
maximum number of unauthenticated connections, 123
message before login, 224
message of the day (MOTD), 153
Microsoft IAS, 199
Microsoft Windows, 30
MSI package, 30
multiple configuration files, 103
multiple host keys, 169

N

network access server (NAS), 146, 199

network address family, 53, 110
Network Address Translation (NAT), 190
network interface binding, 220
network interface card, 64
network logon, 210
network settings, 62
network shares, 230
non-interactive installation, 32

O

OCSP responders, 67, 126, 179
one-way trust, 57, 122
Online Certificate Status Protocol (OCSP), 172, 178
online purchase, 17
OpenSSH authorized_keys file, 84, 143, 176, 300
OpenSSH CA key, 69
OpenSSH certificates, 168
OpenSSH keys, 168, 176
OpenSSH SCP, 93, 96, 156, 157
OpenSSL, removing, 405
OpenSSL cryptographic library, 51
Oracle Solaris, 28
OSS licences, 415
outgoing tunnels, 238
overload
  on CPU, 257

P

pam-account-checking-only, 113
PAM authentication, 145, 193, 193, 194
  upgrading, 196
  with LDAP, 197
PAM library
  upgrading, 196
password
  empty/blank, 174
  expired, 163, 174
password authentication, 84, 88, 144, 145, 173, 175, 193
password cache, 57, 87, 128
  exporting, 58, 310
  importing, 59, 311
PEM encoding, 337
permissions
  configuration file, 258
  host key, 257
PKCS #7 package, 178
planning the installation
  FIPS mode, 15
Pluggable Authentication Module (PAM), 193, 194
pluggable-authentication-modules, 116
port forwarding, 235
  local, 238
  remote, 243
  restricting, 235
port number
  default, 64, 121
private key
  host, , 118, 169, 172
privileged users, 219
problem situations, 257
processes
  maximum number, 50, 123
profile directory, 176
protocol
  Tectia Mapper, 399
proxy scheme, 112
proxy server, 179
proxy settings, 54
public key
  host, , 118
  user, 175
public-key authentication, 84
  server, 60, 118, 168
  user, 142, 175, 247
public-key signature algorithms, 86, 143

Q

quiet-login, 115

R

RADIUS authentication, 88, 146, 193, 199
random_seed file, 299
recording ptyless sessions, 114
Red Hat Linux, 25
registry keys, 39
rekeying interval, 79, 131
related documents, 9
remote administration, 219

remote port forwarding, 243
remote tunnels, 99, 161, 235, 243
removing OpenSSL, 405
removing Tectia Server
from AIX, 33
from HP-UX, 33
from Linux, 34
from Solaris, 34
from Windows, 35
old versions, 19
reporting failed logins, 207
requirements
for disk space, 16
for hardware, 16
for upgrading, 18
resolve client hostname, 53
resolve-client-hostname, 113
restoring default settings, 47
restricting services, 88, 153, 228, 236
restricting tunneling, 235, 236
local tunnels, 96, 158
remote tunnels, 99, 161
with external application, 160, 399
retired servant, 124
revoked certificate, 172
RFC 4253, 322
RFC 4716, 323
rights
log on locally, 175
Rocky Linux, 25
root directory, 176
RPM packages, 25
RSA Authentication Agent, 197
RSA Authentication Server, 197
RSA key
private,
public,
RSA key pair, 169
RSA SecurID, 197

S

SCEP client, 332
secure application connectivity, 235
secure file transfer, 227
Secure File Transfer Protocol (SFTP), 176
Secure Shell server
starting, 41
stopping, 41
secure system administration, 219
SecurID authentication, 88, 145, 193, 197
selector handling rules, 106
selectors, 69, 104
administrator, 75
blackboard, 140
certificate, 71, 136
host certificate, 72, 137
interface, 70, 130, 138
IP, 72, 130, 138
public key passed, 76, 141
user, 73, 138
user group, 74, 139
user password change needed, 142
user privileged, 75, 139
servant lifetime, 50, 50
server
starting, 41
stopping, 41
server authentication
with certificates, 171
with external key, 173
with public key, 168
server authentication methods, 60, 118, 167
server banner message, 224
server certificate, 119, 171
server configuration, 45
server configuration file, 35, 102, 109
server configuration tool, 45
server host key, 35
server settings, 46
server status, 47
services
restricting, 88, 153, 228, 236
settings, default, 47
setting user, 142
setting users to a group, 83, 88, 135, 152
SFTP audit messages, 93, 154
SFTP subsystem, 229
SFTP virtual folders, 94, 230
shared user account, 236
shell access, 219

signature algorithms, 86, 143
silent installation, 32
simple GUI mode, 47
socket, 64
SOCKS server URL, 66, 124
Solaris
  BSM, 226
  installation, 28
  uninstallation, 34
ssh-certview-g3, 336
ssh-cmpclient-g3, 325
  commands, 326
  examples, 330
  options, 327
ssh-ekview-g3, 340
ssh-keyfetch, 321
  environment variables, 323
  examples, 323
  options, 321
ssh-keygen-g3, 170, 315
  examples, 320
  options, 315
ssh-scepclient-g3, 332
  commands, 333
  examples, 335
  options, 333
ssh-server-config.xml, 35, 102, 109
ssh-server-ctl, 304
  commands, 305
  options, 304
ssh-server-g3, 294
  environment variables, 297
  options, 295
ssh-troubleshoot, 256, 313
  commands, 314
  options, 313
starting the server, 41
status, 47
stopping the server, 41
subsystems, 102, 154
  denying audit messages, 154
  executing directly, 154
support, 11
supported operating systems, 15
supported platforms, 15
SUSE Linux, 25
system audit
  on Solaris, 226
system configuration, 45
system log, 47, 65, 122, 223
system requirements, 15

T

technical support, 11
Tectia Client, 13
Tectia ConnectSecure, 13
Tectia Mapper Protocol, 399
Tectia Server, 13
  starting, 41
  stopping, 41
Tectia Server Configuration tool, 13, 45
Tectia Server control utility, 37
Tectia Server for IBM z/OS, 13
terminal access, 91, 153, 219
terminology, 12
ticket forwarding, 86, 147
total number of connections, 50, 124
troubleshooting, 249
  user authentication with certificates, 187
troubleshooting log, 47
troubleshooting tool, 256
trust, one-way, 57, 122
trusted_hosts directory, 189, 299
trusted CA, 69, 127, 178
trusted domain authentication (Windows), 209
tunneling, 235
  access control, 236
  agent, 158, 247
  restricting, 235
  X11, 158, 245
tunnels, 235
  local (outgoing), 96, 158, 235, 238
  remote (incoming), 99, 161, 235, 243

U

Ubuntu, 27
umask, 155
uninstalling Tectia Server
  from AIX, 33
  from HP-UX, 33

from Linux, 34
from Solaris, 34
from Windows, 35
upgrading, 19
use cases, 41
user account
domain, 175, 176
local, 175
shared, 236
user authentication
host-based, 188, 190
with certificates, 178, 178
with certificates (Windows), 181
with GSSAPI, 201
with keyboard-interactive, 193
with password, 173
with public key, 175
user authentication chain, 203
user authentication forwarding, 206, 247
user authentication methods, 80, 133, 167
user configuration directory, 52, 114
user group, 175
user home directory, 94, 230
User Manager, 175
user name
domain policy settings, 55
user name handling on Windows, 208
USERPROFILE, 11
user profile directory, 176
user-specific configurations
on Unix, 37
on Windows, 39
user started processes, 54, 54
UTF-8, 93, 155

V

viewing event log, 47
viewing troubleshooting log, 47
virtual directories, 94
virtual folders, 94, 230

W

well-known port, 235
white list, 50, 129
white list size, 51, 130
Windows
domain, 57, 122
domain precedence, 121
Event Log, 30, 47, 65, 223
installation, 30
logon type, 53
modify installation, 406
password, 173, 175
registry keys, 39
terminal mode, 53
trusted domain authentication, 209
uninstallation, 35
user authentication with certificates, 181
user group, 175
user name, 208
windows-logon-type, 115
windows-terminal-mode, 115
Windows User Manager, 175

X

X.509 certificates, 172, 178
X11 forwarding, 158, 245
x11-listen-address, 112, 246
X11 listener address, 246
XAuth path, 112
XAuth shell, 112
XML attribute
allow-missing, 131, 131, 144, 147
allow-ticket-forwarding, 147
allow-undefined, 107
authorization-file, 143
authorized-keys-directory, 143
chroot, 153, 154, 157
client-nas-identifier, 146
default-path, 115
dir-mask-bits, 134
disable-authorization, 144
disable-crls, 127
discard-limit, 129
dll-path, 145, 145, 147
enable-password-change, 146
failure-delay, 144, 145
http-proxy-url, 124
idle-timeout, 153
ignore-aix-login, 114

ignore-aix-rlogin, 113
ignore-nisplus-no-permission, 115
login-grace-time, 133
mask-bits, 134
max-connections, 123
max-new-connection-queue, 123
max-processes, 123
max-tries, 144, 145
max-unauth-connections, 123
max-unauth-connections-total, 123
openssh-authorized-keys-file, 143
pam-account-checking-only, 113
print-motd, 153
proxy-scheme, 112
quiet-login, 115
record-ptyless-sessions, 114
require-dns-match, 144
resolve-client-hostname, 113
service-name, 145
set-group, 135
signature-algorithm, 143
socks-server-url, 124
tcp-keepalive, 130
total-connections, 124
trusted, 127
use-expired-crls, 127
user-config-dir, 114
white-list-size, 130
windows-domain-precedence, 121
windows-logon-type, 115
windows-terminal-mode, 115
x11-listen-address, 112
xauth-path, 112
xauth-shell, 112
XML DTD, 109, 275
XML element, 109
address-family, 110
attribute, 155
attribute umask, 155
attribute utf8-mode, 155
authentication, 134, 149
authentication-methods, 133
auth-file-modes, 134, 228
auth-gssapi, 147
auth-hostbased, 144
auth-keyboard-interactive, 145
auth-password, 144
auth-publickey, 142
banner-message, 133
ca-certificate, 127
cert-cache-file, 126
cert-validation, 124
cipher, 131
command, 157
connection, 130
connections, 130
crl-auto-update, 127
crl-prefetch, 127
crypto-lib, 110
dod-pki, 127
domain-policy, 121
environment, 153
externalkey, 119
group, 152
hostkey, 117
hostkey-algorithm, 132
ldap-server, 125
limits, 123
listener, 121
load-control, 129
log-events, 122
logging, 122
mac, 131
mapper, 147
mkex, 132
ocsp-responder, 126
openssh-ca-key, 128
params, 110
password-cache, 128
pluggable-authentication-modules, 116
private, 118
protocol-parameters, 117
public, 118
radius-server, 146
radius-shared-secret, 146
rekey, 131
rule, 153
selector, 130, 135, 152
selector/blackboard, 140
selector/certificate, 136

selector/host-certificate, 137
selector/interface, 130, 138
selector/ip, 130, 138
selector/publickey-passed, 141
selector/user, 138
selector/user-group, 139
selector/user-password-change-needed, 142
selector/user-privileged, 139
servant-lifetime, 124
services, 152
set-blackboard, 142
settings, 111
set-user, 142
submethod-aix-lam, 146
submethod-generic, 146
submethod-pam, 145
submethod-password, 145
submethod-radius, 146
submethod-securid, 145
subsystem, 154
terminal, 153
tunnel-agent, 158
tunnel-local, 158
tunnel-local/dst, 159
tunnel-local/mapper, 160
tunnel-local/src, 158
tunnel-local/tunnel-src, 159
tunnel-remote, 161
tunnel-remote/listen, 162
tunnel-remote/src, 161
tunnel-remote/tunnel-dst, 161
tunnel-x11, 158
windows-domain, 122
x509-certificate, 119