SSH.COM

Tectia® Client 7.0

User Manual

27 November 2025

Tectia® Client 7.0: User Manual

27 November 2025

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

Table of Contents

Chapter 1 About This Document

This document describes installing and using Tectia Client. This manual is meant for Tectia Client users and administrators who install and configure the software.

This document contains the following information:

- Installing Tectia Client
- Getting started
- Authentication
- Transferring files
- Tunneling
- Troubleshooting
- Appendices, including command-line tool, GUI, and audit message references

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

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

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**

Convention	Usage	Example
`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:
  - 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, and in the Tectia Server for IBM z/OS client tools. Connection Broker handles all cryptographic operations and authentication-related tasks.

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 Server, and Tectia Server for IBM z/OS (including the Tectia Server for IBM z/OS client tools).

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

PrivX Desktop File Transfer GUI
: PrivX Desktop includes a graphical user interface (GUI) for handling and performing file transfers interactively. The GUI is supported on Windows and Linux.

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 Client

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

2.1 Preparing for Installation

This section lists the supported platforms and the prerequisites for the Tectia Client installation.

2.1.1 System Requirements

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

Table 2.1. Supported operating systems for Tectia Client and Server

Operating System	Client	Server
IBM AIX (POWER)	7.2, 7.3	7.2, 7.3
Oracle Solaris (SPARC)	11	11
Oracle Solaris (x86-64)	11	11
Red Hat Enterprise Linux (x86-64)	8, 9, 10	8, 9, 10
Rocky Linux (x86-64)	8, 9, 10	8, 9, 10
Ubuntu (x86-64)	22.04	22.04
Debian GNU/Linux (x86-64)	12, 13	12, 13
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, Server 2025	10, 11, Server 2016, Server 2019, Server 2022, Server 2025

Note

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

2.1.2 Hardware and Disk Space Requirements

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

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

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

2.1.3 Licensing

Tectia Client requires a license to function. The license file is named `stc70.dat`.

Depending on the platform for which you have purchased Tectia Client, 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 Client software restarted.

2.1.4 Installation Packages

The installation packages of Tectia Client 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 Client bundle:

- For AIX platforms:

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

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

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

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

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

- For Solaris SPARC platform:

```
tectia-client-<version>-solaris-11-sparc-comm-pqc.tar
tectia-client-<version>-solaris-11-sparc-comm.tar
tectia-client-<version>-solaris-11-sparc-upgrd-eval.tar
```

- For Solaris x86-64 platform:

```
tectia-client-<version>-solaris-11-x86_64-comm-pqc.tar
tectia-client-<version>-solaris-11-x86_64-comm.tar
tectia-client-<version>-solaris-11-x86_64-upgrd-eval.tar
```

- For Windows platforms:

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

<version> indicates the product release and the current build number (for example `7.0.0.123`).

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

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

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

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

2.1.5 Upgrading Previously Installed Tectia Client Software

Note

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

If you are running both 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 Client 7.0. 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	Linux	Solaris	Windows
4.x	remove	remove	remove	remove
5.x-6.0	upgrade on top	upgrade on top	remove	remove
6.1-7.0	upgrade on top	upgrade on top	remove	upgrade on top or remove if Transparent TCP Tunneling is installed

The configuration file format and file locations have been changed in Tectia Client 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 7.0 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 7.0 defaults or the PQC hybrid KEX can be enforced.

- The 5.x-6.1 configuration files are used by 7.0 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 7.0, but the default 7.0 configuration is used. However, the connection profiles are migrated from 4.x to 7.0 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.

If you have the Transparent TCP Tunneling option installed, uninstall the previous version of the client before upgrading to the 7.0 Tectia Client and restart the computer after the uninstallation. See Section 2.3.5.

Note

As of version 6.4.18 on Unix, the Transparent TCP Tunneling option is no longer included in the installation package.

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

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

where *<version>* is the Tectia release and *<date>* is the date of the upgrade.

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 Client** 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 7. 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 7.0. Minor releases publish new features and fixes to the previous versions.
   - Maintenance releases are third digit versions, for example 7.0.0. 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 Client below.

2.2 Installing the Tectia Client Software

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

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

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 Client.

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

1. Unpack the downloaded `tar` package.
2. Unpack the installation packages:

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

   In the commands, *<version>* is the current package version of Tectia Client (for example, `7.0.0.123`).
3. 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-client-<version>-aix-6-7-powerpc.bff SSHTectia.Client
   ```
4. Copy the license file to directory: `/etc/ssh2/licenses`. (*This is not necessary in "third-digit" maintenance updates.*) See also Section 2.1.3.

2.2.2 Installing on Linux (RPM)

Tectia Client for Linux platforms is supplied in RPM (Red Hat Package Manager) binary packages for Red Hat Enterprise Linux and SUSE Linux running on the 64-bit x86-64 platform 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 Client. If you want to use the product with a graphical user interface (GUI), install also the optional GUI support package.

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

1. Unpack the downloaded `tar` package.
2. When installing on SUSE or Red Hat Enterprise Linux versions running on the 64-bit x86-64 architecture, use the packages named:

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

In the commands, `<version>` indicates the product release version and the current build number (for example, `7.0.0.123`).

3. Install the packages with root privileges:

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

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

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

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

5. Before you can run the *PrivX Desktop* client GUI, you will need to install the following dependencies:

On SUSE:

```
# zypper install libxcb-cursor0
```

On RHEL/Rocky:

```
# dnf install xcb-util-wm xcb-util-keysyms xcb-util-cursor
```

Note

You need to enable the EPEL repository to obtain some of the dependencies.

Extra dependencies for PrivX Desktop have been verified on RHEL/Rocky 8.x and later, and may need to be adapted for other RPM-based distributions.

2.2.3 Installing on Linux (DEB)

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

The Tectia installation bundle contains the DEB installation files and the license file.

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

1. Download the relevant installation bundle according to your license type:

- Commercial Tectia Quantum Safe Edition License:

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

- Commercial License:

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

- Evaluation:

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

  In the package names, <*version*> corresponds to the release version and build number, for example `7.0.0.123-1`.

2. Unpack the downloaded `tar` package.

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

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

4. Install the packages with root privileges:

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

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

6. Before you can run the *PrivX Desktop* client GUI, install the following dependencies:

   ```
   # apt install libxcb-cursor0
   ```

Note

Extra dependencies for PrivX Desktop have been verified on Ubuntu, and may need to be adapted for other Debian-based distributions.

2.2.4 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 Client.

Tectia Client includes support for Zones on Solaris 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, if the local zones are added into the system later, the Tectia Client needs to be separately installed on them.

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

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

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

1. Unpack the downloaded `tar` package.

2. When installing on Solaris version 11 running on the SPARC architecture, use the packages named:

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

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

   ```
   ssh-tectia-common-<version>-solaris-11-x86_64.pkg.Z
   ssh-tectia-client-<version>-solaris-11-x86_64.pkg.Z
   ```

   In the commands, *<version>* indicates the product release version and the current build number (for example, `7.0.0.123`).

3. Unpack the installation packages to a suitable place. The standard place is `/var/spool/pkg` in Solaris environment. In the command examples below, we use Solaris 11 x86-64:

   ```
   $ uncompress ssh-tectia-common-<version>-solaris-11-x86_64.pkg.Z
   $ uncompress ssh-tectia-client-<version>-solaris-11-x86_64.pkg.Z
   ```

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

   ```
   # pkgadd -d ssh-tectia-common-<version>-solaris-11-x86_64.pkg all
   # pkgadd -d ssh-tectia-client-<version>-solaris-11-x86_64.pkg all
   ```

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

2.2.5 Installing on Windows

The Windows installation packages are provided in the MSI (Windows Installer) format for Microsoft Windows versions running on the 64-bit (x86-64) platform architecture.

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

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

Note

If you want to install both Tectia Client and Tectia Server on the same machine, you must install both products using the Tectia Server installer `ssh-tectia-server-<version>-windows-<platform>.msi`, where *<version>* shows the Tectia Client/Server release version and build number (for example `7.0.0.123`), and *<platform>* shows the platform architecture (`x86_64` for 64-bit Windows versions).

If you are upgrading a previous installation of Tectia Client, see first Section 2.1.5.

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

1. Extract the installation zip file contents to a temporary location.
2. Locate the correct Windows Installer file `ssh-tectia-client-<version>-windows-<platform>.msi`, where:
   - `<version>` shows the Tectia Client/Server release version and build number, for example `7.0.0.123`.
   - `<platform>` shows the platform architecture `x86_64` for 64-bit Windows versions.
3. Double-click the installation file, and the installation wizard will start.

Note

The license file 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 install the `stc70.dat` license file after completing the installation. The installation wizard will show an error message about missing license file (see below), and when you attempt to start the Tectia Client, you are prompted to install the license manually to the correct directory:

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

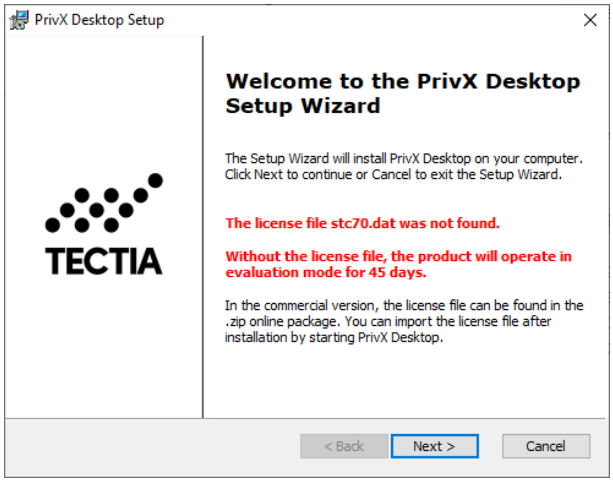


Figure 2.1. Warning about a missing license file

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**.

4. Follow the wizard through the installation steps and fill in information as requested.
5. The **Typical** installation of Tectia Client includes the **sshg3.exe**, **scpg3.exe**, and **sftpg3.exe** command-line tools, and the graphical user interface for terminal and file transfer.

To also install Transparent TCP Tunneling (described in Section 6.1.1), select **Complete** when the wizard prompts for the setup type.

To select specific components to install, select **Custom** when the wizard prompts for the setup type. The next dialog box allows you to exclude some of the components from the installation. See Figure 2.2.

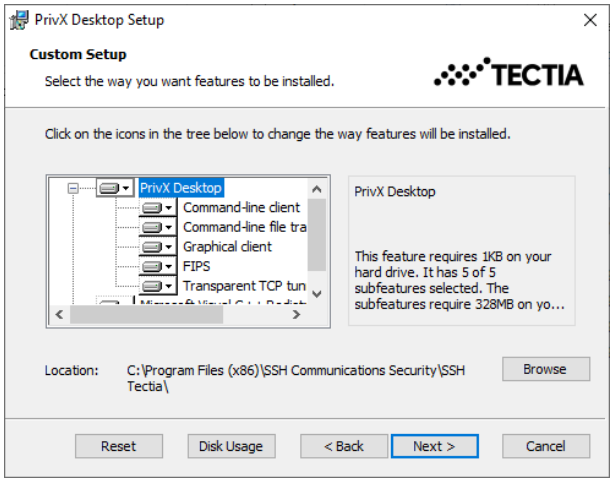


Figure 2.2. Installation options with Tectia Client

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

7. You have to restart the computer after installing Tectia Client. Click **Yes** to restart.

The default installation directory is:

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

Silent Installation

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

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

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

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

In the command:

- *<version>* shows the current version of Tectia Client, for example `7.0.0.123`.

- *<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 Client is installed to the default location.

Desktop Icon

During Windows installation, a PrivX Desktop icon is added to your desktop. It opens the GUI for Tectia Client.

Figure 2.3. The PrivX Desktop GUI icon

2.3 Removing the Tectia Client Software

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

Note

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

2.3.1 Removing from AIX

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

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

```
# installp -u SSHTectia.Client
```

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

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

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

2.3.2 Removing from Linux (RPM)

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

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

```
# rpm -e ssh-tectia-client
```

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

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

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

```
# rpm -e ssh-tectia-guisupport
```

2.3.3 Removing from Linux (DEB)

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

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

```
# dpkg -P ssh-tectia-client
```

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

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

2.3.4 Removing from Solaris

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

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

```
# pkgrm SSHG3clnt
```

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

```
# pkgrm SSHG3cmmn
```

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

2.3.5 Removing from Windows

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

Using Windows Control Panel tools

1. From the Windows **Start** menu, open the **Control Panel** and click **Programs and Features**.

2. Select **Tectia Client** from the list of installed programs and click **Uninstall**.

3. Click **Yes** to confirm.

Using the Windows Installer

1. Locate the Windows Installer file `ssh-tectia-client-<version>-windows-<platform>.msi`, where:

   - `<version>` shows the Tectia Client/Server release version and build number, for example `7.0.0.123`.
   - `<platform>` shows the platform architecture `x86_64` for 64-bit Windows versions.

   On some Windows versions the `.msi` file type is not shown for the installer file.

2. Double-click the installer file, and the Windows Installer will start.

3. Select **Remove** to start the uninstallation.

4. Click **Finish** when the removal has been completed.

Using Silent Command-line Tools

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

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

In the command, `<version>` is the version of Tectia Client to be removed (for example, `7.0.0.123`), and `<platform>` shows the platform architecture (`x86_64` for 64-bit Windows versions).

2.4 Files Related to Tectia Client

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

2.4.1 File Locations on Unix

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

- `/etc/ssh2`
  - `/etc/ssh2/ssh-broker-config.xml`: the global Connection Broker configuration file (see ssh-broker-config(5))
  - `/etc/ssh2/ssh-broker-config-example.xml`: a sample file with Connection Broker configuration examples
  - `/etc/ssh2/licenses`: the license file directory (see Section 2.1.3).
  - `/etc/ssh2/hostkeys`: the global directory for known remote host keys
- `/opt/tectia/share/auxdata/ssh-broker-ng`: the Connection Broker configuration file DTD directory
  - `/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd`: the document type definition (DTD) used with the Connection Broker configuration files. **Do not edit** this file!

- `/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`: this configuration file is read first, and it holds the factory default settings. **Do not edit** this file, but you can use it to view the default settings. This file must be available and correctly formatted for the Connection Broker to start. For the configuration options, see ssh-broker-config(5).

Note

In Tectia Client 6.1 and earlier on Unix the default auxiliary data directory `auxdata` was located in `/etc/ssh2/ssh-tectia/`. If your `ssh-broker-config.xml` file was created for Tectia Client version 6.1 or earlier, please update its DOCTYPE declaration to contain the current path to the Connection Broker configuration file DTD directory: `/opt/tectia/share/auxdata/ssh-broker-ng/`.

- `/opt/tectia/bin`: user binaries such as **sshg3** and **ssh-broker-g3**
- `/opt/tectia/man`: the manual pages
- `/opt/tectia/libexec`: library binaries
- `/opt/tectia/lib/sshsecsh`: library binaries

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

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

2.4.2 File Locations on Windows

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

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

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

- `"<INSTALLDIR>\SSH Tectia Client"`: the binaries for Tectia Client
- `"<INSTALLDIR>\SSH Tectia Broker"`: the Connection Broker binaries and example configuration files

- "`<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml`": the global Connection Broker configuration file (see its manpage: ssh-broker-config(5))
- "`<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config-example.xml`": a sample file with Connection Broker configuration examples

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

Figure 2.4 shows the Tectia directory structure in the Windows Start menu when several Tectia products have been installed on the same machine.

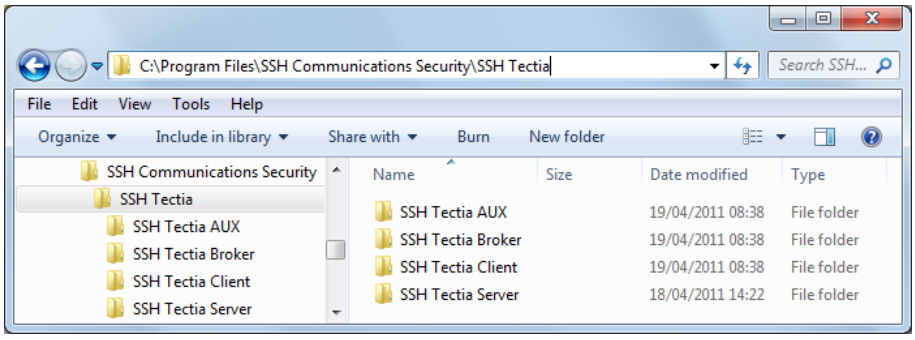


Figure 2.4. The Tectia directory structure on Windows

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

- `%APPDATA%\SSH\ssh-broker-config.xml`: the user-specific Connection Broker configuration file with connection profiles

- `%APPDATA%\SSH\global.dat`: the 4.x-6.x version Tectia SSH Terminal GUI configuration file
- `%APPDATA%\SSH\*.ssh2`: the 4.x-6.x version Tectia Terminal GUI profile configuration files
- `%APPDATA%\SSH\random_seed`: the seed file for the random number generator
- `%APPDATA%\SSH\HostKeys`: the user-specific directory for known remote host keys
- `%APPDATA%\SSH\UserKeys`: the default directory for user public-key pairs
- `%APPDATA%\SSH\UserCertificates`: the default directory for user certificate key pairs
- `%APPDATA%\SSH\identification`: (*optional*) the identification file used with public-key authentication

Note

The user-specific `%APPDATA%` directory is hidden by default. To view hidden directories, change the setting in Windows Explorer. For example, select **Organize → Folder and search options** on the menu. On the **View** tab, under **Hidden files and folders**, select **Show hidden files, folders and drives**.

2.4.3 Registry Keys on Windows

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

- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia Broker`
- `HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application\SSH Tectia Broker GUI`
- `HKLM\SOFTWARE\SSH Communications Security\SSH Tectia`
- `HKLM\SOFTWARE\SSH Communications Security\SSH Tectia Client`
- `HKLM\SOFTWARE\Wow6432Node\SSH Communications Security\SSH Tectia` (on x64 architecture, only)
- `HKLM\SOFTWARE\Wow6432Node\SSH Communications Security\SSH Tectia Client` (on x64 architecture, only)

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

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

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

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

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

sshg3/scpg3/sftpg3

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

ssh/scp/sftp

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

Chapter 3 Getting Started with Tectia Client

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

3.1 Product Components

Tectia Client consists of the following components:

- The Connection Broker: **ssh-broker-g3**, **ssh-broker-ctl**
- Secure Shell command-line tools: **sshg3**, **scpg3**, **sftpg3**
- Auxiliary command-line tools: **ssh-keygen-g3**, **ssh-cmpclient-g3**, **ssh-scepclient-g3**, **ssh-certview-g3**, **ssh-ekview-g3**

On Linux, macOS and Windows there are additional GUI components:

- PrivX Desktop GUI (see Section 3.2.1)
- PrivX Desktop File Transfer GUI (see Section 5.2)
- Tectia Connections Configuration GUI (see Section A.1)

3.2 First Login to a Remote Host

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

There are separate instructions on using the PrivX Desktop GUI to connect to a remote server host (see Section 3.2.1) and on using `sshg3` on the command line (see Section 3.2.2).

Tectia Client includes a shortcut menu that helps configuring the connection settings. description of the Tectia shortcut menu, see Section A.5.

3.2.1 Logging in with PrivX Desktop GUI

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

You can connect to a remote host by using PrivX Desktop as follows:

1. Open the Tectia Client GUI for example by clicking the PrivX Desktop icon on your desktop.

2. PrivX Desktop offers several ways to open a Secure Shell connection:

   - On **Quick Connect** page you can provide host name or profile name and change the most common connection settings for this connection without editing the profile.

   - If you already have an ongoing session, open a **New Connection** in a new tab by clicking the **+** button. You can connect to a new remote host computer and still keep the old connection to a different host open.

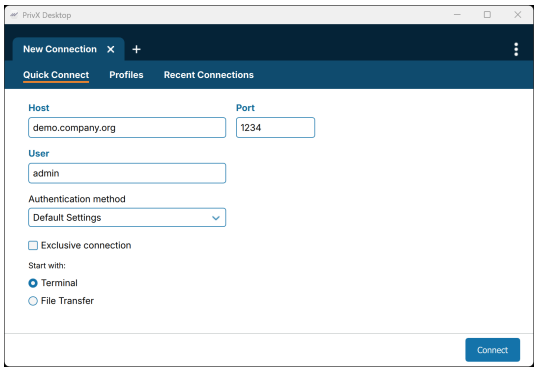
   - If earlier session has been disconnected, you can open a new session by pressing **Enter** or **Space** on the keyboard when the (still disconnected) terminal or file transfer window is active.

   - If you or the administrator has defined connection profiles, you can also connect from **Profiles** page of the **New Connection** tab by clicking the profile tile. From the alternate profile tile menu you can choose to `Open Terminal` or `Open File Transfer` initially instead of the default channel type.

     In this case, the settings defined in the profile (hostname, port, user name etc.) are automatically used for the connection.

   - On **Recent Connections** page of the **New Connection** tab, you can quickly reopen connections to the remote hosts that have been disconnected and closed since the Connection Broker has been started or open additional connections to the host with ongoing connections.

3. On **Quick Connect** page of the **New Connection** tab you can define the server host you want to connect to:



Define at least the Host and click **Connect**:

- **Host** – the FQDN, short host name, the IP address of the remote host or the connection profile name.
- **User** – your user name on the remote host.
- **Port** – specify alternate port number if the default Secure Shell listener port 22 is not used on the remote host.
- **Authentication Method** – by default enabled user authentication methods from `Default Settings` are used unless you specify one of the Quick Connect user authentication methods. `Password` will attempt methods used commonly for password-based methods like Password and Keyboard-interactive only, and `Public-Key` will use available certificates and public keys only.
- **Exclusive connection** – by default a new connection to the same remote host will open a new channel. To open an additional secure shell connection or prevent opening additional channels later within the connection, enable Exclusive connection checkbox.
- **Start with** – `Terminal` or `File Transfer` as the initial channel in the secure shell connection. You can later open additional channels or new connections to the server.

Environment variables can be used to pre-fill the Quick Connect values. If undefined, the Default Settings will be used and the client prompts for required values if needed.

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

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

   When public-key authentication is used to authenticate the server, *the first connection is very important*. When Tectia Client receives a new server host key, it will display the host identification message.

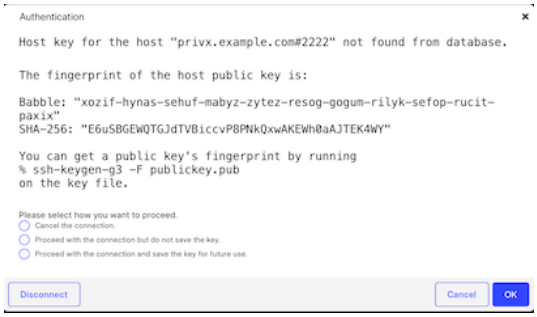


Figure 3.1. The host identification dialog – the first connection to a remote host

The message displays the fingerprint of the host's public key in the SSH Babble format that is a series of pronounceable five-letter words in lower case and separated by dashes. By default also base64-encoded SHA-256 fingerprint is shown.

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

   Caution

   Never save a host public key without verifying its authenticity!

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

   Information on the server public key will be stored on the client-side machine so that the client can later validate the key. On Tectia Client, the public key information is stored in the user's hostkeys directory:

   `$HOME/.ssh2/hostkeys`

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

   For more information on server authentication, see Section 4.2.

7. The user authentication phase starts. You will be prompted to authenticate yourself to the server using the authentication method you selected in the **Connect to Server** dialog, or by default with your password or with the passphrase of your private key. The required authentication method depends on the server settings.

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

3.2.2 Logging in with Command-Line sshg3

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

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

```
$ sshg3 <hostname>
```

For example:

```
$ sshg3 abc.example.com
```

The basic syntax is:

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

where:

- `user` - Enter a user name that is valid on the remote host. The `user@` attribute is optional. If no user name is given, the local user name is assumed.
- `host` - Enter the name of the remote host as an IP address, FQDN (fully qualified domain name), or short host name. The remote host must be running a Secure Shell version 2 server.
- `port` - Enter the number of the Secure Shell listen port on the remote server. The `#port` attribute is optional. If no port is given, the default Secure Shell port 22 is assumed.

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

```
$ sshg3 profile1
```

In this case, the settings defined in the profile (host name, port, user name etc.) are used for the connection. For instructions on creating and editing the connection profiles, see the section called “The `profiles` Element”.

For more information on the **sshg3** commands and options, see sshg3(1).

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

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

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

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

```
$ sshg3 user@server
Host key for the host "server" not found from database.

The fingerprint of the host public key is:

Babble: "xozif-hynas-sehuf-mabyz-zytez-resog-gogum-rilyk-sefop-rucit-paxix"
SHA-256: "E6uSBGEWQTGJdTVBiccvP8PNkQxwAKEWh0aAJTEK4WY"

You can get a public key's fingerprint by running
% ssh-keygen-g3 -F publickey.pub
```

```
on the key file.

Please select how you want to proceed.
    cancel) Cancel the connection.
      once) Proceed with the connection but do not save the key.
      save) Proceed with the connection and save the key for future use.
Please select one (cancel, once, save):
```

The message shows the fingerprint of the host's public key in the SSH Babble format that is a series of pronounceable five-letter words in lower case and separated by dashes and by default in base64-encoded SHA-256 format.

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

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

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

   For more information on server authentication, see Section 4.2.

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

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

3.3 Using Public-Key Authentication

Public-key authentication is based on the use of digital signatures. To use public-key authentication, you must first create a key pair on the client, and upload the public key to the server. For instructions, see Section 4.5.

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

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

3.4 Configuring Tectia Client

Tectia Client includes a default configuration that can get you started. To tailor the Tectia Client behaviour according to the needs of your environment, you can edit the existing configuration.

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

On Linux, macOS and Windows, Tectia Client provides a graphical user interface for handling the Connection Broker configuration. On other platforms, the configuration can be edited directly in the configuration file (in XML format). The Connection Broker settings can be edited using the Tectia Connections Configuration GUI that can be started for example from the Tectia Client tray icon menu.

3.4.1 Connection Broker Configuration

For users of Tectia Client, the most relevant and most typically needed item to configure for the Connection Broker are the connection profile settings. All other settings are typically configured by system administrators.

It is advisable to create connection profiles for servers where you will need to connect repeatedly. The profiles contain the server ID, your user name on that server, and information on the authentication method to be used.

In general, the following aspects can be configured for the Connection Broker:

Secure connection details

These settings define how Tectia Client will establish the secure connections to the remote servers, for example: what type of a connection will be opened, what authentication methods will be applied, will a proxy be used and is tunneling allowed.

User and server authentication methods

The user authentication settings define the methods Tectia Client will use when sending user authentication data to the remote servers. The Tectia Connections Configuration GUI includes a public-key wizard (on Linux and Windows) that helps in creating and uploading public keys to the servers.

The server authentication settings define how the remote servers will be authenticated by Tectia Client.

Tunneling of connections

Tunnels can be defined to secure all or some TCP applications and FTP connections. It is also possible to allow forwarding of X11 sessions and SSH connections from one remote server to another.

Tip

The first things to configure are the user authentication settings (creating public keys for the users and uploading them to remote servers) and creating connection profiles for servers where you will need to connect repeatedly.

For instructions on defining the authentication settings, see Chapter 4, and for the authentication-related options in the configuration file, see **authentication-methods** .

For instructions on creating connection profiles via the GUI, see Section A.1.3, and about adding connection profiles directly into the configuration file, see the section called "The `profiles` Element".

For a detailed description of the Connection Broker **configuration options**, see Appendix A.

3.4.2 Connection Broker Configuration Files

The Connection Broker configuration is stored in an XML file named `ssh-broker-config.xml`. You can edit the configuration file with your favorite XML or text editor, but make sure `ssh-broker-config.xml` remains a valid XML file. For details about the Connection Broker configuration options, see ssh-broker-config(5).

When you want to modify the Connection Broker configuration, you will typically edit a user-specific copy of the configuration file stored in `$HOME/.ssh2` on Unix, `%APPDATA%\SSH\` on Windows). You need to create the user-specific configuration file first.

The Tectia Connections Configuration GUI on Windows and Linux also writes to the user-specific configuration file automatically.

For a list of the related configuration files and their locations:

- on Unix, refer to Section 2.4.1
- on Windows, refer to Section 2.4.2

3.4.3 Command-Line Tools

Tectia Client includes command-line tools **sshg3**, **scpg3** and **sftpg3** that can be used to open secure connections and to transfer files securely - the same as with the PrivX Desktop GUI

These tools can be used in scripts and in real-time with a set of options detailing their behaviour. The options given on command line will override the settings specified in the configuration file.

The options of each command-line tool are described on the man pages sshg3(1), scpg3(1), and sftpg3(1)).

3.5 Creating Connection Profiles

On Tectia Client on Windows and Linux, you can configure separate connection settings for each Secure Shell server you connect to. You can also create several profiles for the same server, for example, with different user accounts.

On Windows, you can add connection profiles via the following views:

- Start PrivX Desktop GUI and open the **Tectia Connections Configuration GUI** by selecting **Configuration** from the menu icon.

(Alternatively, you can open the Tectia Connections Configuration GUI by right-clicking the PrivX Desktop tray icon in the Windows taskbar notification area and selecting **Configuration** from the shortcut menu.)

On Linux, open the **Tectia Connections Configuration GUI**:

1. Go to the `/opt/tectia/bin` directory by entering:

```
$ cd /opt/tectia/bin/
```

2. Start the **Tectia Connections Configuration GUI** with the following command:

```
$ ssh-tectia-configuration
```

In the Tectia Connections Configuration GUI, go to the **Connection Profiles** page (as shown below) and click **Add profile**.

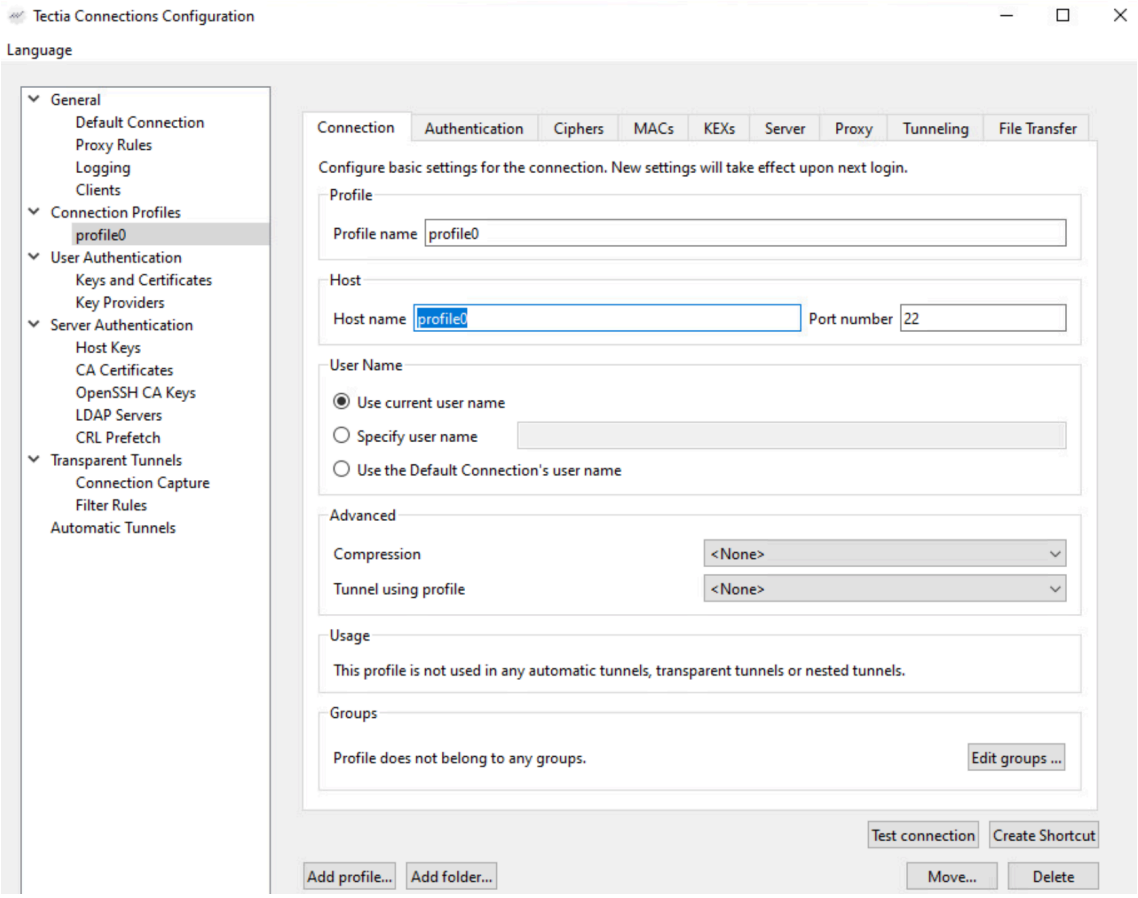


Figure 3.2. Adding connection profiles

Newly created connection profiles will inherit the default values for authentication, ciphers, MACs, KEXs, tunneling, and advanced server settings defined under the **General → Default Connection** page. The values can be customized on the profile-specific tabbed pages, see Figure 3.3.

To rename a connection profile, right-click the profile name in the **Connection Profiles** list and click **Rename**. Type in the new name.

To remove a connection profile, select the profile and click **Delete**. You will be asked for confirmation. Click **Yes** to proceed with the deletion.

3.5.1 Defining Connection Profile Settings

Under the **Connection Profile** page, on the **Connection** tab, you can define the protocol settings used in the connection. Any changed connection settings will take effect the next time you log in.

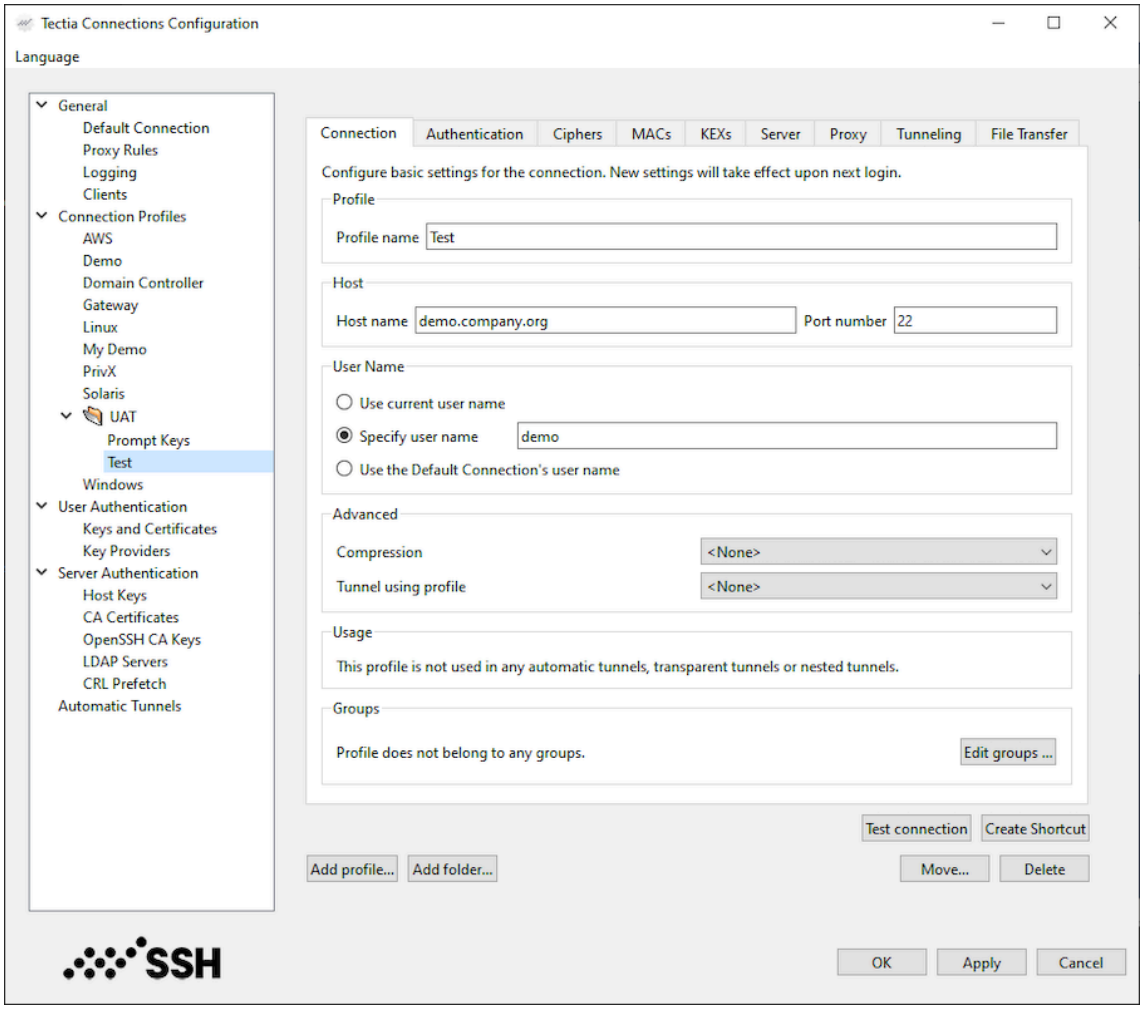


Figure 3.3. Configuring connection profiles

Profile

In **Profile name**, type a name for the profile.

Host

In **Host name**, enter the name of the remote host computer to which you want to connect with the profile.

In **Port number**, enter the port number you want to use for the Secure Shell connection. The default port is 22.

Note

A Secure Shell server program must be listening to the specified port on the remote host computer or the connection attempt will not succeed. If you are unsure which port the remote host computer is listening to, contact the system administrator of the remote host.

User Name

Select **Use current user name** if the connection should always be made using the currently logged in Windows or Unix user name. This is similar to defining `%USERNAME%` (note the percent signs) as the user name.

Select **Specify user name** and enter the user name, if you want to define the user name to be used when connecting to the remote host computer. If you specify `%USERNAME%` (note the percent signs) as the user name, it will be replaced with the name of the current Windows or Unix user account upon connecting.

Advanced

Not needed now: In **Compression**, select the desired compression setting from the drop-down menu. Valid choices are **zlib** and **none**. Compression is disabled by default.

Not needed now: In **Tunnel using profile**, select the desired connection profile from the drop-down menu. Any nested tunnels will be created through the profile. For information on the tunneling features, refer to Chapter 6.

3.6 Enabling FIPS 140-2 Mode

You can enable Tectia Client to operate in FIPS mode after which all cryptographic operations are run according to the FIPS 140-2 standard.

In FIPS mode, OpenSSL cryptographic libary is used for all cryptographic operations, see Section 3.6.3. In Standard mode, Tectia proprietary cryptographic library is used for all cryptographic operations.

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.

3.6.1 Enabling FIPS Mode Using Configuration GUI

To enable FIPS mode on Windows:

1. Open Tectia Connections Configuration GUI (see Section A.1.1).

2. Go to the General settings by selecting **General** in the tree view.
3. Under **Cryptographic Library**, select **FIPS mode**.
4. Ensure that the cryptographic algorithms defined for the default connection settings or any connection profile are compatible with FIPS mode. You will be informed of algorithms that are not allowed in FIPS mode. For FIPS-compatible algorithms, see Appendix E.
5. Click **Apply**.
6. Click **Stop Broker** from the Tectia shortcut menu (see Section A.5).
7. Start a new client or connection that launches a new Connection Broker in FIPS mode.

Note

On Windows, you can switch all Tectia products to FIPS mode by creating a file named `FIPSMODE` in the `SSH Tectia AUX` folder. Note that while the `FIPSMODE` file is present, all Tectia products will be in FIPS mode regardless of their configurations the next time they are restarted.

On Windows with Tectia Server also installed on the same machine as Tectia Client, this file is created and removed automatically when FIPS mode is changed with the Tectia Server Configuration GUI and configuration is applied.

3.6.2 Enabling FIPS Mode Using Configuration File

To enable FIPS mode on Unix:

1. Open the Connection Broker configuration file `ssh-broker-config.xml` that you want to modify (see the section called “Connection Broker Files”.
2. Under the `general` element, modify the `crypto-lib` element by settings its value to `fips`.
3. Ensure that the cryptographic algorithms defined in the configuration file for the `default-settings` element and the `profiles` element are compatible with FIPS mode. For FIPS-compatible algorithms, see Appendix E.
4. Save the configuration file and stop the Connection Broker if it is running:

```
$ ssh-broker-ctl stop
```

5. Start a new connection. You may then verify the new Connection Broker is running in FIPS mode with:

```
$ ssh-broker-ctl status
```

Note

On Unix, you can switch all Tectia products to FIPS mode by creating a file named `/etc/ssh2/FIPSMODE`. Note that while the `FIPSMODE` file is present, all Tectia products will be in FIPS mode regardless of their configurations the next time they are restarted.

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
```

3.6.3 FIPS-Certified Cryptographic Library

Tectia products 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 Client. This OpenSSL FIPS-certified cryptographic library is used to provide the classes of functions listed in the following tables.

The functions from the OpenSSL 3.0.12 24 Oct 2023 (FIPS provider: 3.0.9) used on Linux, Windows, and Solaris are listed in Table 3.1.

Table 3.1. APIs used from the OpenSSL cryptographic library version 3.0

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.

Chapter 4 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 user. Both parties are assured of the identity of the other party.

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

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

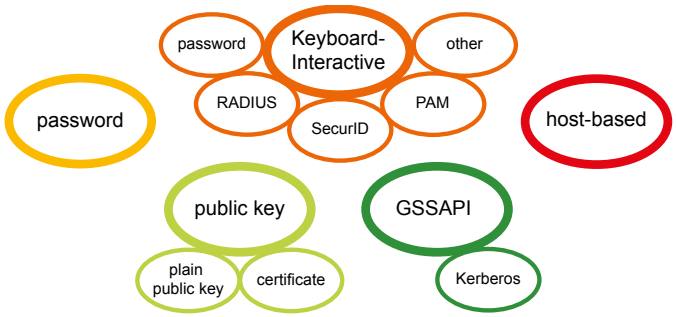


Figure 4.1. User authentication methods

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

When several interactive authentication methods are defined as allowed, Tectia Client will alternate between the methods and offers each of them in turn to the server in case the previous method failed. This makes it possible to define different authentication methods for different users, and they can be handled with the same server configuration.

4.1 Supported User Authentication Methods

The following user authentication methods are supported in the Tectia client/server solution.

Table 4.1. User authentication methods supported by the Tectia client/server solution

Authentication method	Tectia Server		Tectia Client	
	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.

4.1.1 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)

4.2 Server Authentication with Public Keys

The server is authenticated with a digital signature based on an RSA, DSA, ECDSA, or Ed25519 public-key algorithm. At the beginning of the connection, the server sends its public key to the client for validation.

Server authentication is done during Diffie-Hellman key exchange through a single public-key operation. When public-key authentication is used to authenticate the server, the first connection is very important. During the first connection the client will display a message similar to the one in Figure 4.2.

Figure 4.2. Tectia Client on Windows – first connection to a remote host

Caution

Never save a host public key without verifying its authenticity!

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

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

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

The stored information on the host keys is used in subsequent connections to those remote hosts. Tectia Client checks which type of a host key (DSA, RSA, ECDSA or Ed25519) it possesses for a particular server, and automatically chooses the key exchange algorithm to be used in the connection between the client and server accordingly. This makes it quicker to connect to hosts for which only one type of host key has been stored.

When **`auth-server-publickey`** is set to some other policy than `strict` (as it is by default), if logging is enabled for the Connection Broker, Tectia Client will log information about changed and new host public keys with their fingerprints in the syslog (on Unix) or Event Viewer (on Windows).

4.2.1 Host Key Storage Formats

When the host key is received during the first connection to a remote host (or when the host key has changed) and you choose to save the key, its file name is stored in hashed format, `keys_hhh...`, where

`hhh` is a hash of the host port and name. The saved file contains a hash of the host's public key. A salt is included in the hash calculations. The value of the salt is stored in the file `salt` in the same directory as the host keys (`$HOME/.ssh2/hostkeys` on Unix, `%APPDATA%\SSH\HostKeys` on Windows). The hashed host key format is a security feature to make address harvesting on the hosts difficult.

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

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

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

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

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

Note that the host identification is different based on the host name and port the client is connecting to. The host name can occur in four different formats:

- Fully qualified domain name (FQDN)
- Short host name
- IPv4 address
- IPv6 address

The host key for each name format has to be saved separately, as they are not mutually exchangeable.

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

- `sshg3 user@alpha`

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

- `sshg3 user@alpha.example.com`

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

- `sshg3 user@10.1.101.10`

  produces the key with IPv4 address (in plain format `key_22_10.1.101.10.pub`)

- `sshg3 user@fd00:10:1:103::1:2f69`

  produces the key with IPv6 address (in plain format `key_22_fd00010000101030000000000012f69.pub`)

When connecting to a server using its IPv6 address, the IPv6 address given to Tectia Client is canonicalized without the colons, and the canonical format is used in the known host key file name. For example, the plain format host key file for `::1#10022` would be `key_10022_00000000000000000000000000000001.pub`. The canonical format is also used in the process of saving and reading hashed host keys.

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

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

4.2.2 Using the System-Wide Host Key Storage

If a host key is not found in the user-specific host key directory, it is next searched on Unix from the `/etc/ssh2/hostkeys` directory, on Windows from the `C:\ProgramData\SSH\HostKeys` directory. Host key files are not automatically put in the system-wide directory but they have to be updated manually by the system administrator (`root`).

The process for distributing the host keys manually is explained in the following. The instructions reflect the Unix file paths but are applicable also to Windows. Simply replace the Unix paths with the corresponding Windows paths.

Storing Keys in the Hashed Format

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

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

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

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

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

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

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

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

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

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

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

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

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

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

In this case, because of `--overwrite=no`, if a key for server 'alpha' already exists, the command will fail and the key will not be updated.

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

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

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

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

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

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

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

Storing Keys in the Plain Format

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

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

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

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

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

On the client:

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

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

4.2.3 Resolving Hashed Host Keys

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

In Tectia Connections Configuration GUI, the tool is available on the **Host Keys** page. See the section called “Managing Host Keys”.

On the command line, the command syntax is:

```
ssh-keygen-g3 -F host_name[#port]
```

For example:

```
ssh-keygen-g3 -F examplehost#222
```

The *host_name* can be the fully qualified domain name, short host name, or the IP address of the remote host. The *port* definition is optional in the command. If no port is given, the default Secure Shell port 22 is assumed.

The tool shows the location, fingerprint (in the SSH babble format) and type (RSA, DSA, ECDSA or Ed25519) of the requested host's public key or keys. For example:

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

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

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

You can define several file locations to be checked for host keys. For more information, see Section 4.2.4.

4.2.4 Using the OpenSSH `known_hosts` File

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

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

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

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

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

4.3 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.

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

1. The server sends its certificate (which contains a public key) to the client. The packet also contains random data unique to the session, signed by the server's private key.

2. As the server certificate is signed with the private key of a certification authority (CA), the client can verify the validity of the server certificate by using the CA certificate.

3. The client checks that the certificate matches the name or the IP address of the server. When endpoint identity check is enabled in the Connection Broker configuration (either in the `ssh-broker-config.xml` file with the **`cert-validation`** attribute `end-point-identity-check`, or in the Tectia Connections Configuration GUI **CA Certificates** page with the **Enable endpoint identity check** option) the client compares the server's host name or IP address to the Subject Name or Subject Alternative Name (DNS Address) specified in the server certificate.

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

Caution

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

Endpoint identity check can also be configured to make Tectia Client ask the user to either accept or cancel the connection if the server's host name does not match the one in the certificate.

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.

4.3.1 Managing CA Certificates with the Configuration File (Unix)

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

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

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

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

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

```
<cert-validation end-point-identity-check="yes"
                 http-proxy-url="http://proxy.example.com:800">
  <ldap-server address="ldap://ldap.example.com:389" />
  <ocsp-responder url="http://ocsp.example.com:8090" validity-period="0" />
  <dod-pki enable="no" />
  <ca-certificate name="ssh_ca1"
                  file="ssh_ca1.crt"
                  disable-crls="no"
                  use-expired-crls="100" />
</cert-validation>
```

   The client will only accept certificates issued by the defined CA(s) or its subordinate CA(s). Typically, Authority Info Access URI from the server certificate is used to construct a validation path to the

trusted CA. It is also possible to configure LDAP Server or add the intermediate CA certificate(s) to the user config directory as extra certificates if AIA does not exist in the certificates themselves.

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

Note

CRL usage should only be disabled if the CA issues short-lived certificates or for testing purposes. Otherwise it is highly recommended to always use CRLs.

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

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

4.3.2 Managing CA Certificates with the GUI

Using the Tectia Connections Configuration GUI to manage CA certificates is described in the section called “Managing CA Certificates”.

4.4 User Authentication with Passwords

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

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

On Windows, password authentication uses the Windows password to authenticate the user at login time. Also, if the SSH server allows it, users with administrator privileges may retain their permissions by adding `elevated,` before their user name. For example:

```
$ sshg3 elevated,Administrator@example.com
```

4.4.1 Defining Password Authentication with the Configuration File (Unix)

To enable password authentication on the client, the `authentication-methods` element of the `ssh-broker-config.xml` file must contain an `auth-password` element:

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

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

4.4.2 Using Stored Passwords in Connection Profiles

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

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

```
<profiles>
  <profile>
    <authentication-methods>
      <auth-password />
    </authentication-methods>
  ...
    <password file="path/to/file" />
  </profile>
...
</profiles>
```

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

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

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

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

Caution

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

Note

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

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

Caution

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

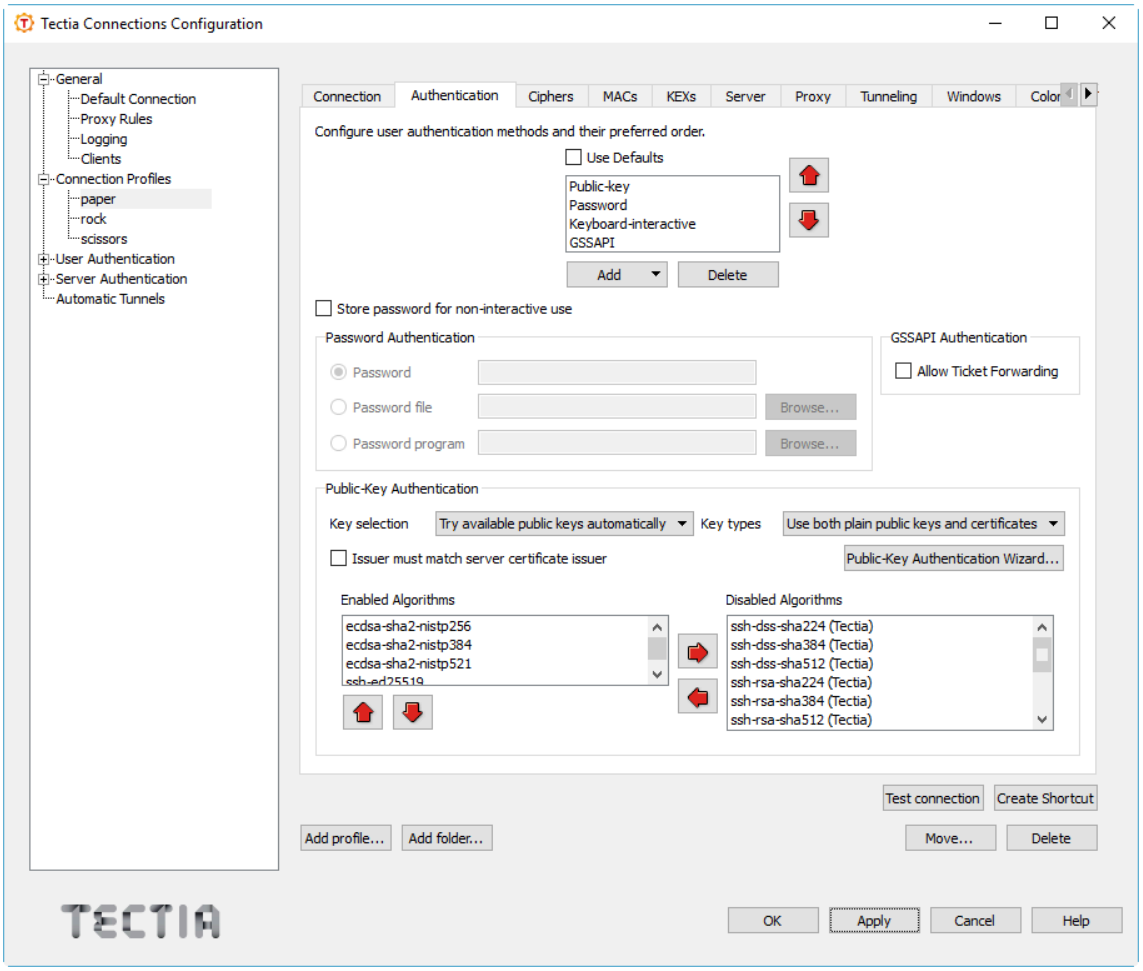


Figure 4.3. Configuring authentication methods for the profile

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

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

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

Note

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

4.4.3 Managing Authentication Methods with the GUI

Using the Tectia Connections Configuration GUI to manage authentication methods is described in the section called "Defining Authentication".

4.5 User Authentication with Public Keys

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

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

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

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

Caution

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

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

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

1. Generate a key pair. You can generate your own key files with the help of a built-in Public-Key Authentication Wizard on Windows (see Section 4.5.3), or with **ssh-keygen-g3** on Unix or Windows command line (see Section 4.5.1).

   You can also import existing keys on the **Keys and Certificates** page of the Tectia Connections Configuration GUI. See the section called "Managing Keys and Certificates".

2. Upload your public key to the remote host computer. On Windows, you can do this automatically (see the section called "Uploading Public Keys Automatically"). On Unix and Windows, you can also copy the public key manually (see Section 4.5.2).

In the instructions in the following sections,

- `Server` is the remote host running the Secure Shell server that you are trying to connect to.
- `ServerUser` is the user name on `Server` that you are logging in as.
- `Client` is the host running the Secure Shell client (Tectia Client).
- `ClientUser` is the user name on `Client` that should be allowed to log in to `Server` as `ServerUser`.

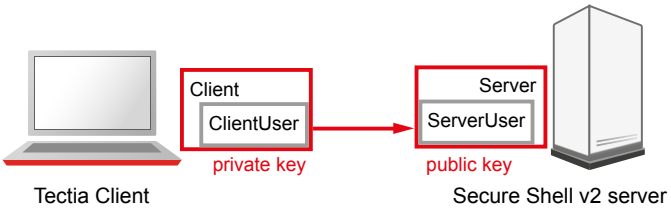


Figure 4.4. User public-key authentication

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

4.5.1 Creating Keys with ssh-keygen-g3

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

```
$ ssh-keygen-g3
Generating 3072-bit rsa key pair
   9 oOo.oOo.oOo
Key generated.
3072-bit rsa, ClientUser@Client, Mon Aug 15 2022 12:08:07 +0200
Passphrase :
Again :
Private key saved to  /home/ClientUser/.ssh2/id_rsa_3072_a
Public key saved to   /home/ClientUser/.ssh2/id_rsa_3072_a.pub
```

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

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.

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

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

In the example above, the private key file is `id_rsa_3072_a`. The public key file is `id_rsa_3072_a.pub`, and it can be distributed to other computers.

By default, **ssh-keygen-g3** creates a 3072-bit RSA key pair. DSA, ECDSA or Ed25519 keys can be generated by specifying the `-t` option with **ssh-keygen-g3**. Key length can be specified with the `-b` option. For automated jobs, the key can be generated without a passphrase with the `-P` option, for example:

```
$ ssh-keygen-g3 -t ecdsa -b 384 -P
```

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

4.5.2 Uploading Public Keys Manually

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

To enable public-key authentication with your key pair:

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

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

   You can also add other directory locations for keys on the **Keys and Certificates** page of the Tectia Connections Configuration GUI. See the section called “Managing Keys and Certificates”. On Unix, you can use the `general/key-stores/key-store` element in the `ssh-broker-config.xml` file. See the section called “Key Store Configuration Examples”.

2. (*Optional*) Create an identification file.

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

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

   Edit it with your favorite text editor to include the following line (replace `id_rsa_3072_a` with the file name of the private key):

   ```
   IdKey       id_rsa_3072_a
   ```

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

   ```
   IdKey       UserKeys\id_rsa_3072_a
   ```

   The identification file can contain several key IDs. For more information on the syntax of the identification file, see `$HOME/.ssh2/identification` .

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

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

   - Tectia Server on Unix or z/OS:

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

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

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

   - Tectia Server on Windows:

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

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

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

   - OpenSSH server on Unix or z/OS:

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

4. Copy the public key to `Server`.

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

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

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

     ```
     $ scpg3 id_rsa_3072_a.pub ServerUser@tectia_server:.ssh2/authorized_keys/
     ```

   - Tectia Server 4.x on Unix and Windows:

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

     ```
     $ scpg3 id_rsa_3072_a.pub ServerUser@tectia4x_server:.ssh2/
     ```

   - Tectia Server for IBM z/OS:

     The public key must be converted to the EBCDIC format. This can be done by including the **scpg3** `dst-site` command-line option, or the **sftpg3 site** commands in the file transfer command. For more information on the **site** parameters, see Section 5.3.1.

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

     ```
     $ scpg3 --dst-site="C=ISO8859-1,D=IBM-1047,X=TEXT" id_rsa_3072_a.pub \
     ```

```
ServerUser@tectia_zos:$HOME/.ssh2/
```

- OpenSSH server on Unix:

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

  ```
  $ scpg3 id_rsa_3072_a.pub ServerUser@open_unix:.ssh/
  ```

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

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

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

   - Tectia Server 4.x on Unix and Windows and Tectia Server for IBM z/OS require an `authorization` file stored in the user's `.ssh2` directory. The authorization file specifies the public keys that are authorized for login.

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

     ```
     $ sshg3 ServerUser@tectia_server4x "echo Key id_rsa_3072_a.pub >> \
     .ssh2/authorization"
     ```

     On a Windows server:

     ```
     $ sshg3 ServerUser@tectia4x_win "cmd /c echo Key id_rsa_3072_a.pub >> \
     .ssh2\authorization"
     ```

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

     ```
     Key     id_rsa_3072_a.pub
     ```

     This directs Tectia Server to use `id_rsa_3072_a.pub` as a valid public key when authorizing your login.

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

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

     ```
     $ sshg3 ServerUser@open_server "ssh-keygen -i -f id_rsa_3072_a.pub >> \
     .ssh/authorized_keys"
     ```

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

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

```
</authentication-methods>
```

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

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

Try to log in:

```
Client$ sshg3 Server
```

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

4.5.3 Creating Keys with the Public-Key Authentication Wizard

On Windows and Linux, you can use the Tectia **Public-Key Authentication Wizard** to generate a key pair and to upload a public key to a host, see the section called "Public-Key Generation" and the section called "Uploading Public Keys Automatically". The wizard will generate two key files, your private key and your public key.

The new private and public key will be stored on your local computer in the `%APPDATA%\SSH\UserKeys` directory on Windows and in the `$HOME/.ssh2/` directory on Linux. The private key file has no file extension, and the public key has the same base file name as the private key, but with `.pub` as the file extension.

Make sure that public-key authentication is allowed in the Connection Broker configuration, in the default settings and in the relevant connection profile (it is allowed by default). For the default settings, see the section called "Defining Authentication", and for the connection profile, see the section called "Defining Authentication".

To use the key pair for public-key authentication, you have to upload the public key to the remote host computer. If the remote host has an SFTP server running, you can automatically upload a copy of your new public key to the server with the wizard. To upload the key automatically, see the section called "Uploading Public Keys Automatically". To upload the key manually, see Section 4.5.2.

Public-Key Generation

New keys are generated in the Tectia Connections Configuration GUI. Under **User authentication**, select the **Keys and Certificates** page and click **New Key** to start the **Public-Key Authentication Wizard**.

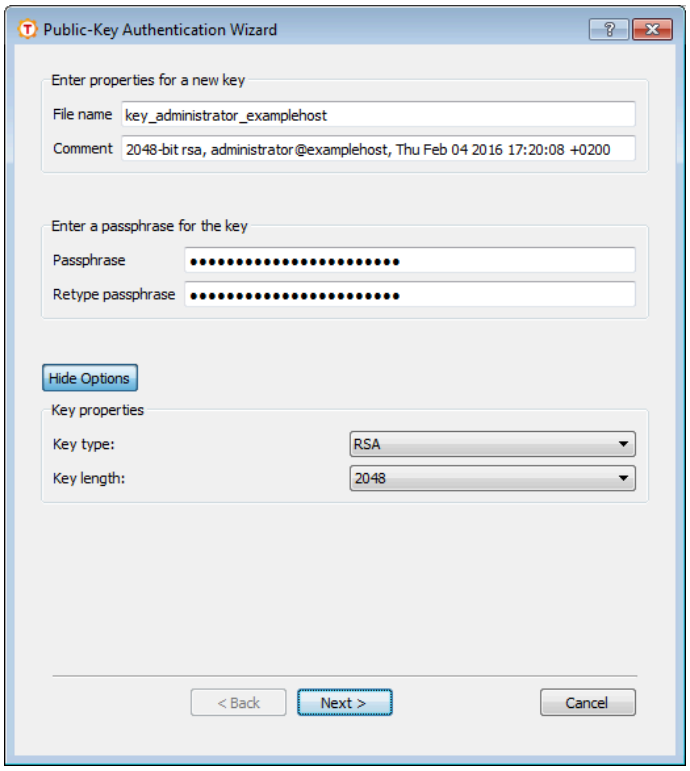


Figure 4.5. The Public-Key Authentication Wizard

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

File name

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

Comment

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

Passphrase

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

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.

Make the passphrase difficult to guess. Use ideally at least 20 characters, both letters and numbers. Any punctuation characters can be used as well. While the passphrase or private key is never sent over the network, a dictionary attack can be used against a private key if it is accessible locally. For ease

of use, an authentication agent is recommended instead of leaving the passphrase empty. By default ssh-broker-g3 functions as an authentication agent.

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

Retype passphrase

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

Click **Advanced Options** to define the type of the key to be generated and the key length to be different from the defaults. By default, Tectia Client generates a pair of 3072-bit RSA keys.

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

Key type

Select the type of the key to be generated. Available options are Ed25519 RSA (default), ECDSA and DSA.

Note

In FIPS mode (conforming to FIPS 186-5) RSA, ECDSA and Ed25519 are supported. DSA has been deprecated.

Key length

Select the length (complexity) of the key to be generated. Available options are:

- DSA/RSA keys: 2048, 3072, 4096, 5120, 6144, 7168, 8192 bits
- ECDSA keys: 256, 384, 521 bits
- Ed25519 keys: 256 bits

Larger keys of the same key type are more secure, but also slower to generate. A 256-bit ECDSA key and a 3072-bit RSA key provide equivalent security.

Click **Next** to proceed to uploading the key as instructed in the section called “Uploading Public Keys Automatically”.

Uploading Public Keys Automatically

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

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

Select a key from the **Key and Certificate List** and click **Upload**.

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

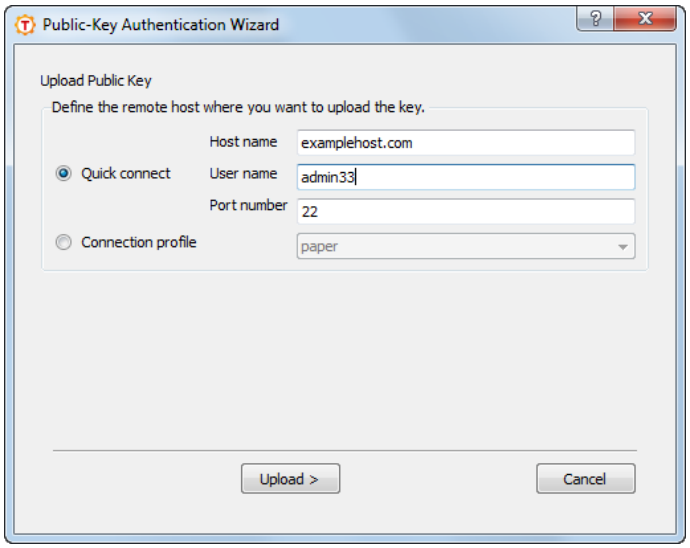


Figure 4.6. Uploading a key

Quick connect

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

Connection profile

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

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

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

Note

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

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

If you do not use the automatic upload facility, see Section 4.5.2.

4.5.4 Using Keys Generated with OpenSSH

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

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

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

You can add OpenSSH keys and directories on the **Keys and Certificates** page of the Tectia Connections Configuration tool. See the section called "Managing Keys and Certificates".

The public key can be uploaded to the server the same way as with standard SSH2 keys. See Section 4.5.2 and the section called "Uploading Public Keys Automatically".

4.5.5 Special Considerations with Windows Servers

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

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

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

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

4.6 User Authentication with Certificates

Certificate authentication is technically a part of the public-key authentication method. The signature created with the private key and the verification of the signature using the public key (contained in the X.509 certificate when doing certificate authentication) are done identically with conventional public keys and certificates. The major difference is in determining whether a specific user is allowed to log in with a specific public key or certificate. With conventional public keys, every server must have every user's public key, whereas with certificates the users' public keys do not have to be distributed to the servers - distributing the public key of the CA (self-signed certificate) is enough.

In brief, certificate authentication works as follows:

1. The client sends the user certificate (which includes the user's public key) to the server. The packet also contains data unique to the session and it is signed by the user's private key.

2. The server uses the CA certificate (and external resources as required) to check that the user's certificate is valid.

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

4. The server matches the user certificate against the rules in the server configuration file to decide whether login is allowed or not.

4.6.1 Using the Configuration File (Unix)

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

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

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

   ```
   $ ssh-cmpclient-g3 INITIALIZE
   -P generate://ssh2:passphrase@rsa:3072/user_rsa \
   -o /home/user/.ssh2/user_rsa -p 62154:ssh \
   -s 'C=FI,O=SSH,CN=user;email=user@example.org' \
   -S http://fw.example.com:1080 http://pki.example.com:8080/pkix/ \
   'C=FI, O=SSH, CN=Test CA 1'
   ```

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

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

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

3. (*Optional*) Create an identification file.

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

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

   ```
   CertKey     user_rsa
   ```

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

   For more information on the syntax of the identification file, see `$HOME/.ssh2/identification`.

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

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

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

4.6.2 Configuring User Authentication with Certificates on Windows

You can configure user authentication with X.509 certificates on Windows using Tectia Connections Configuration GUI. You also need to configure Tectia Server for user authentication with certificates, see *Tectia Server Administrator Manual*.

1. Launch Tectia Connections Configuration GUI.

   Right-click  in the notification area of the Windows taskbar and select **Configuration**.

2. Under **General**, click **Default Connection**. Select the **Authentication** tab. Ensure that public-key authentication is enabled and it is the first or only method in the list. By default, it is enabled.

   Under **Public-Key Authentication**, you can select to use public keys or certificates or both in the authentication.

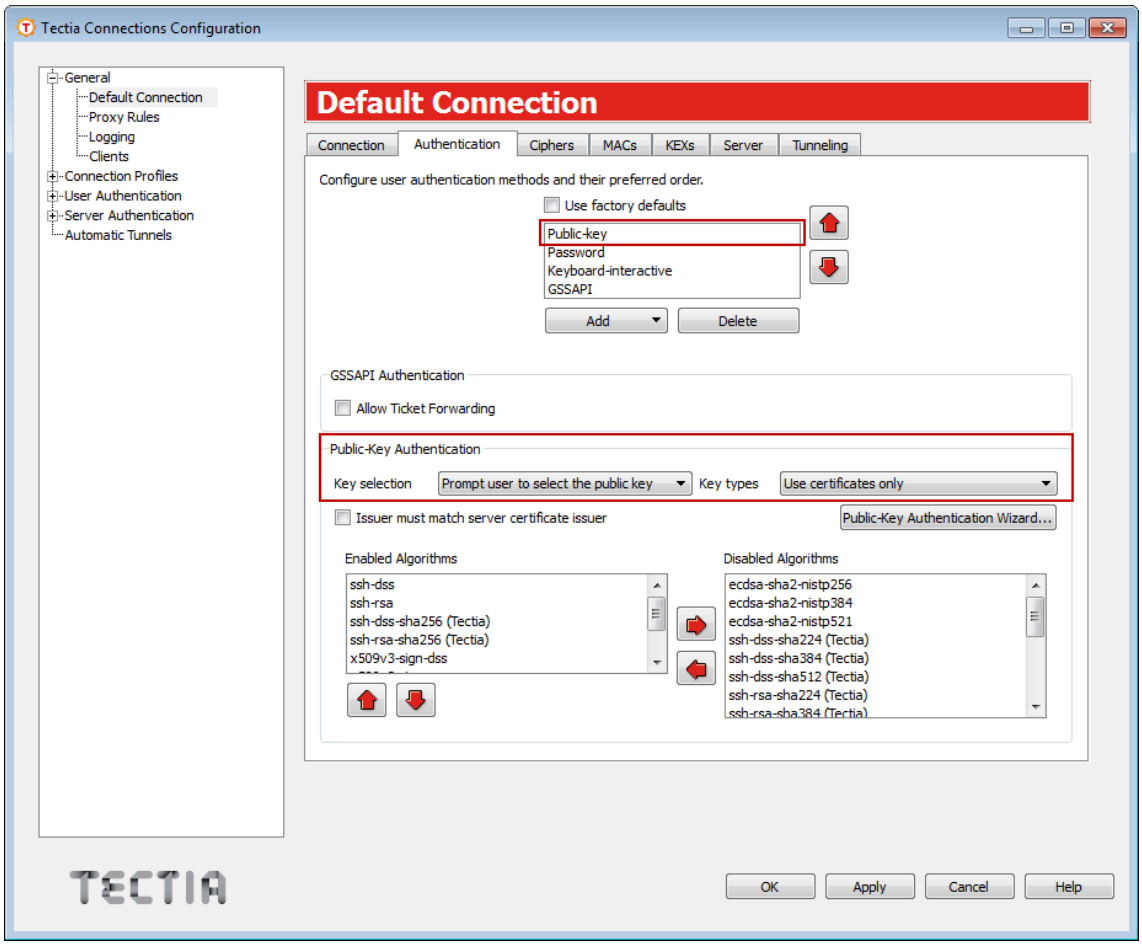


Figure 4.7. Enabling public-key authentication

3. If you are using connection profiles, select the profile name under **Connection Profiles**. Select the **Authentication** tab and ensure that public-key authentication is enabled.

4. Tectia suggests installing the certificate into the Microsoft Certificate store that is a personal store for the user.

5. Under **User Authentication**, select **Key Providers**. Enable **Microsoft Crypto API** and click **Apply**.

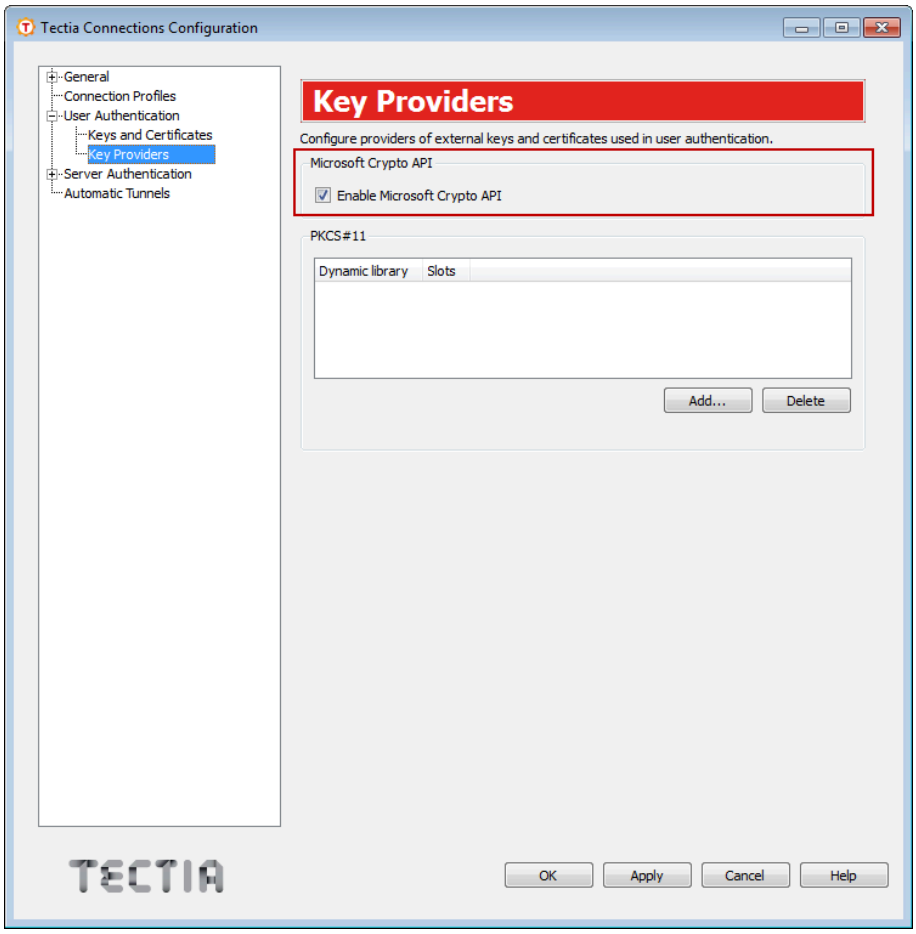


Figure 4.8. Enabling Microsoft Crypto API as a certificate provider

You can also read certificate information from USB tokens or smartcards via Microsoft Crypto API if they are compatible with the API. Alternatively USB tokens or smartcards can be used by enabling **PCKS#11**.

6. The certificate is now loaded into the client automatically. Under **User Authentication**, select **Keys and Certificates**. You can see the available certificates under **Key and Certificate List**.

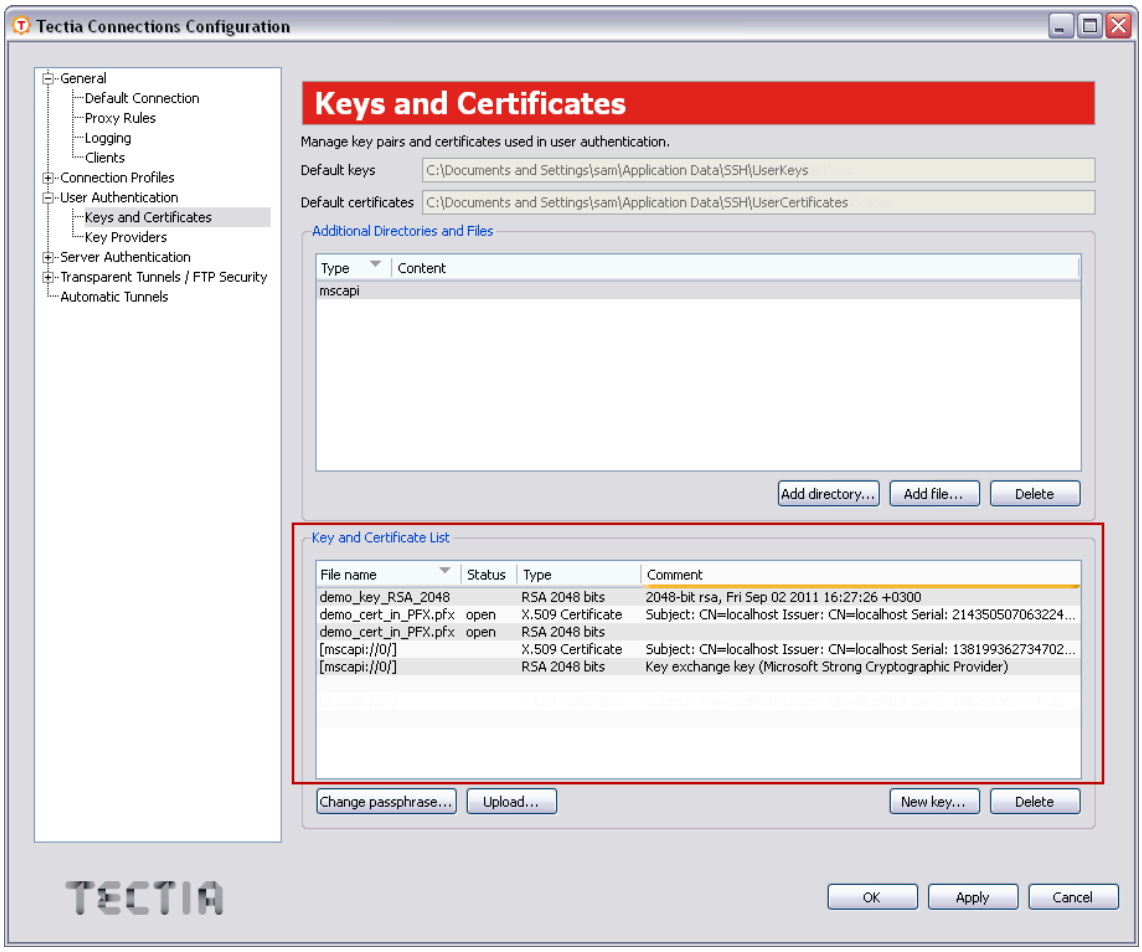


Figure 4.9. Viewing available certificates

Tectia Client can also read key and certificate information from the file system. These can be defined under **Additional Directories and Files**.

Note

Ensure that the client certificate is set up for client authentication only. It makes troubleshooting several certificates easier, for example, as server authentication certificates cannot be used as user certificates.

For more information about the key and certificate settings, see the section called “Managing Keys and Certificates”.

Troubleshooting User Authentication with Certificates

If the certificate authentication does not succeed for some reason, running Tectia Server in the troubleshooting mode and viewing the troubleshooting log can provide a lot of information about the end-user connection. For more information, refer to Section *Starting Tectia Server in Debug mode on Windows* in the *Tectia Server Administrator Manual*.

4.6.3 Importing PKCS Certificates with Tectia Connections Configuration GUI

You can import existing PKCS #12, PKCS #7 and X.509 certificates on the **Keys and Certificates** page under User Authentication in the Tectia Connections Configuration GUI. See the section called "Managing Keys and Certificates".

4.7 Host-Based User Authentication (Unix)

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

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

4.8 User Authentication with Keyboard-Interactive

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

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

Note

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

4.8.1 Defining Keyboard-Interactive Method with the Configuration File (Unix)

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

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

4.8.2 Defining Keyboard-Interactive Method with the GUI

Using keyboard-interactive authentication is a Connection Broker setting. Using the Tectia Connections Configuration GUI to manage authentication methods is described in the section called “Defining Authentication”.

4.9 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.

On Linux platforms, Kerberos libraries are installed by default. 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 client debug log.

4.9.1 Defining GSSAPI Method with the Configuration File (Unix)

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

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

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

4.9.2 Defining GSSAPI Method with the GUI

Using the Tectia Connections Configuration GUI to manage authentication methods is described in the section called “Defining Authentication”.

Chapter 5 Transferring Files

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

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

5.1 Secure File Transfer with scpg3 and sftpg3 Commands

Tectia Client provides commands **scpg3** (secure copy) and **sftpg3** for secure file transfer. These command-line clients apply the Secure File Transfer Protocol (SFTP).

When files are being uploaded with commands **scpg3** and **sftpg3**, the files have the `TRUNCATE` flag on. The file size is shown as 0 until the file transfer has been completed.

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

In case the **scpg3** and **sftpg3** command-line clients are used in scripts that start several file transfer commands at the same time, the Connection Broker must already be running in the background. Since the Connection Broker takes a few seconds to become up and running, make sure the scripts are not started immediately, because they can fail if the Connection Broker is still starting.

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

5.1.1 Using scpg3

scpg3 (**scpg3.exe** on Windows) is used to securely copy files over the network. **scpg3** uses **ssh-broker-g3** to provide a secure transport using the Secure Shell version 2 protocol. The remote host(s) must be running a Secure Shell version 2 server with the **sftp-server** (or **sft-server-g3**) subsystem enabled.

The basic syntax of **scpg3** is:

```
scpg3 user@source:/directory/file user@destination:/directory/file
```

scpg3 can be used to copy files in either direction; from the local system to the remote system or vice versa. Copies between two remote hosts are also permitted. Local paths can be specified without the *user@system:* prefix. Relative paths can also be used, they are interpreted in relation to the user's home directory.

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

```
scpg3 localfile user@destination:/C:/directory/file
```

For more information on the command-line options, see scpg3(1).

5.1.2 Using sftpg3

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

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

The basic syntax of **sftpg3** is:

```
sftpg3 user@host
```

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

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

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

5.1.3 Enhanced File Transfer Functions

The following enhanced file transfer features are available with the **scpg3** and **sftpg3** command-line tools of Tectia Client:

- Checkpoint/restart for transferring large files (with any IETF-compliant SSH server)
- Prefix for ensuring that a file is fully transferred before it is used (with any IETF-compliant SSH server)
- Streaming for improved file transfer speed (with Tectia Servers)

For information on the commands, see scpg3(1) and sftpg3(1).

5.2 Secure File Transfer GUI

Tectia Client provides a secure file transfer GUI that makes it easy to download files from a remote host computer into your local computer and to upload files to a remote host.

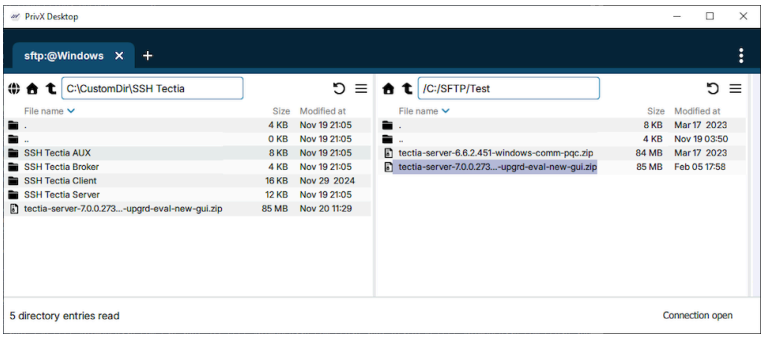


Figure 5.1. PrivX Desktop File Transfer GUI

In the PrivX Desktop File Transfer GUI, you can use the connection profiles defined in the Connection Broker configuration, or connect to a remote host using the **Quick Connect** option.

5.2.1 Downloading Files with PrivX Desktop File Transfer GUI

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

Keyboard shortcut

Select file(s) or a directory with Shift + arrow keys and press `Ctrl+S` to transfer the file or directory recursively to the other side. You can also use Ctrl+A to select all including directory entries or Ctrl +Shift+A to select all files.

Drag and drop

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

Shortcut menu

When you right-click a file in the Remote View, a shortcut menu appears. Select **Download** to perform the transfer.

5.2.2 Uploading Files with PrivX Desktop File Transfer GUI

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

Keyboard shortcut

Select file(s) or a directory with Shift + arrow keys and press `Ctrl+S` to transfer the file or directory recursively to the other side. You can also use Ctrl+A to select all including directory entries or Ctrl +Shift+A to select all files.

Drag and drop

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

Shortcut menu

When you right-click a file in the Local View a shortcut menu appears. Select the **Upload** to perform the transfer.

5.2.3 File Properties and Preview

Selecting a file in the Local View or Remote View and pressing Ctrl+I toggles Information dialog which allows you to view file information including SHA-2 hash and preview the file in various formats.

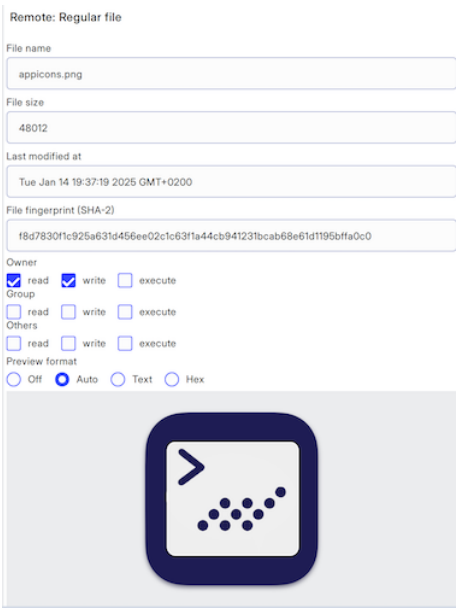


Figure 5.2. Properties and Preview page for a file

5.2.4 Differences from OS tools

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

Caution

Delete in secure file transfer is immediate and permanent operation.

Deleting folders

Deleting a folder will delete the files and subfolders recursively immediately instead of requiring user confirmation or moving the files to Bin as the native operating system tools typically do.

Multiple paste operations

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

Note

The maximum size of transferred files is limited only by the file system. (On many systems the maximum file size is 2 gigabytes.) However, downloading large files with drag and drop is limited as user-specific configuration directory is used temporarily by the PrivX Desktop File Transfer GUI. Use keyboard shortcut or shortcut menu in File Transfer View to download large files.

5.3 Controlling File Transfer

The current Secure File Transfer Protocol (SFTP) does not transfer any information about the files to be transferred, only the file contents as a byte stream. This is sufficient for Unix-type files if the sender and receiver use the same CCS. However, it is possible to set specific file permissions for transferred files by using the **chmod** command with command line tools.

With MVS data sets, Tectia needs more information: which transfer format to use, what code sets are involved, and what the file characteristics are. Tectia introduces some extensions to SFTP and the information can be relayed by using the Site commands of **scpg3** and **sftpg3**.

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

5.3.1 Site Command

For command descriptions, see the **site** and **lsite** command in sftpg3(1) and the `--dst-site` and `--src-site` options in scpg3(1).

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

```
sftp> site x=bin
sftp> site transfer_mode=bin
```

The available **site** parameters are listed and described in the following.

Table 5.1. site parameters

Parameter	Abbreviations	Possible values
AUTOMOUNT	-	YES|NO|IMMED
[NO]AUTOMOUNT	[NO]AUTOM	-
AUTORECALL	-	YES|NO
[NO]AUTORECALL	[NO]AUTOR	-
BLKSIZE	B, BLOCKSI	*size*
BLOCKS	BL	-
CONDDISP	CO	CATLG|UNCATLG|KEEP|DELETE
CYLINDERS	CY	-
DATACLAS	DA	*class*
DATASET_SEQUENCE_NUMBER	SEQNUM	*number*
DEFER	DE	YES|NO
[NO]DEFER	-	-
DIRECTORY_SIZE	M, DI, DIRSZ	*size*
EXPIRY_DATE	EXPDT	*yyddd* | *yyyyddd*
FILE_STATUS	STATUS	NEW|MOD|SHR|OLD
FILETYPE	FILET	SEQ|JES
FIXRECFM	FI	*length*
JOB_ID	JESID	*ID*
JOB_OWNER	JESO	*name*
JOBNAME	JESJOB	*name*
KEYLEN	KEYL	*length*
KEYOFF	KEYO	*offset*
LABEL_TYPE	LABEL	NL|SL|NSL|SUL|BLP|LTM|AL|AUL
LIKE	-	*like*
LRECL	R, LR	*length*
MGMTCLAS	MG	*class*
NORMDISP	NOR	CATLG|UNCATLG|KEEP|DELETE
PRIMARY_SPACE	PRI	*space*
PROFILE	P, PROF	*profile*
RECFM	O, REC	*recfm*
RECORD_TRUNCATE	U, TRUN	YES|NO
[NO]TRUNCATE	[NO]TRU, [NO]TRUN	-
RETENTION_PERIOD	RET	*days*
SECONDARY_SPACE	SE, SEC	*space*

Parameter	Abbreviations	Possible values
SIZE	L	*size*
SPACE_RELEASE	RLSE	YES|NO
SPACE_UNIT	SU	BLKS|TRKS|CYLS|AVGRECLEN
SPACE_UNIT_LENGTH	SUL	*length*
STAGING	S, STAGE	YES|NO
STORCLAS	ST	*class*
SVC99_TEXT_UNITS	SVC99	*string*
TRACKS	TR	-
TRAILING_BLANKS	TRAIL	YES|NO
[NO]TRAILINGBLANKS	[NO]TRAI, [NO]TRAIL	-
TRANSFER_CODESET	C, CODESET	*codeset*
TRANSFER_FILE_CODESET	D, FCODESET	*codeset*
TRANSFER_FILE_LINE_DELIMITER	J, FLDELIM	UNIX|MVS|MVS-FTP|DOS|MAC|NEL
TRANSFER_FORMAT	F, FORMAT	LINE|STREAM|RECORD
TRANSFER_LINE_DELIMITER	I, LDELIM	UNIX|MVS|MVS-FTP|DOS|MAC|NEL
TRANSFER_MODE	X, MODE	BIN|TEXT
TRANSFER_TRANSLATE_DSN_TEMPLATES	A, XDSNT	*templates*
TRANSFER_TRANSLATE_TABLE	E, XTBL	*table*
TYPE	T	PS|PO|PDS|POE|PDSE|GDG| HFS|VSAM|ESDS|KSDS|RRN
UNIT	UN	*unit*
UNIT_COUNT	UC, UNC	*number*
UNIT_PARALLEL	UNP	YES|NO
VOLUME_COUNT	VC, VOLCNT	*number*
VOLUMES	VO, VOL	*vol1+vol2+...*

`AUTOMOUNT=YES|NO|IMMED`

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

If set to `NO`, offline data sets are not mounted automatically.

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

Default: `NO`

`[NO]AUTOMOUNT|[NO]AUTOM`

`AUTOMOUNT|AUTOM` is equal to `AUTOMOUNT=YES`.

`NOAUTOMOUNT|NOAUTOM` is equal to `AUTOMOUNT=NO`.

`AUTORECALL=YES|NO`

Defines whether data sets migrated by a storage manager are recalled automatically.

Default: `YES`

`[NO]AUTORECALL|[NO]AUTOR`

`AUTORECALL|AUTOR` is equal to `AUTORECALL=YES`.
`NOAUTORECALL|NOAUTOR` is equal to `AUTORECALL=NO`.

`BLKSIZE|B|BLOCKSI=` *`size`*

Specifies the maximum block size.

Default: none

`BLOCKS|BL`

Specifies that the space allocation unit is blocks. Equal to `SPACE_UNIT=BLKS`.

`CONDDISP|CO=CATLG|UNCATLG|KEEP|DELETE`

Specifies the disposition of the output file when a file transfer ends prematurely (the client or server are alive but disconnected from the other end; for example, when pressing **CTRL+C** in the client).

Note

If the client (when transferring to local or client side) or the server (when transferring to remote or server side) dies, they will have no control over the disposition.

The options have the following effects, depending on the file type (MVS or HFS):

- `CATLG`: an MVS data set is retained and its name is cataloged. An HFS file is retained.
- `UNCATLG`: the name of an MVS data set is removed from the catalog but the data set is retained. An HFS file is retained.
- `KEEP`: an MVS data set is retained (if cataloged it will be still cataloged, if uncataloged it will be still uncataloged). An HFS file is retained.
- `DELETE`: the name of an MVS data set is removed from the catalog and the space allocated for the data set is released. An HFS file is deleted.

Default: `CATLG`

`CYLINDERS|CY`

Specifies that the space allocation unit is cylinders. Equal to `SPACE_UNIT=CYLS`.

`DATACLAS|DA=` *`class`*

Specifies the data class of a data set.

Default: none

`DATASET_SEQUENCE_NUMBER|SEQNUM=` *`number`*

Identifies the relative position of a data set on a tape volume.

Default: System default

`DEFER|DE=YES|NO`

Specifies whether data set allocation is postponed from allocation phase to when the data set is opened.

If set to `YES` data set allocation is postponed until data set is opened.

If set to `NO` data set is allocated in allocation phase.

Default: `NO`

`[NO]DEFER|DE`

`DEFER|DE` is equal to `DEFER=YES`.
`NODEFER` is equal to `DEFER=NO`.

`DIRECTORY_SIZE|M|DI|DIRSZ=` *`size`*

Specifies the number of 256-byte records in the directory.

Default: `10`

`EXPIRY_DATE|EXPDT=` *`yyddd|yyyyddd`*

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

Default: System default

`FILE_STATUS|STATUS=NEW|MOD|SHR|OLD`

Defines the status of a data set. If entered, the value will be used when allocating the data set. This attribute corresponds to the first value in the `DISP` parameter of the JCL DD statement. Possible values are:

- `NEW`: Create a data set.
- `MOD`: Append to an existing data set. If the data set does not exist, a new data set is created.
- `SHR`: Create a read-only data set.

- `OLD`: Designate an existing data set.

`FILETYPE|FILET=SEQ|JES`

Specifies whether to interface with the file system or with the z/OS Job Entry Subsystem (JES).

Using `FILETYPE=JES` enables the commands **put** and **sput** to submit transferred files to the internal reader job queue for execution, and **get** and **sget** commands to retrieve spool data sets. To terminate interfacing with JES and return to normal file access, set the file type back to sequential (SEQ), or to an empty string (that is, `FILETYPE=`). Entering an empty string as file type sets the file type to default.

Default: `SEQ`

`FIXRECFM|FI=` *`length`*

The data set organization is set to `FB` and the fixed record length is set to *`length`*.

Default: none

`JOB_ID|JESID=` *`ID`*

When in `FILETYPE=JES` mode, `JOB_ID` specifies that commands accessing the JES spool, such as **get**, apply only to jobs with a job ID that matches the supplied *`ID`*.

Commands **get**, **sget**, and so on, with a job ID can be used to retrieve the spool files for a given job.

`JOB_OWNER|JESO=` *`name`*

When in `FILETYPE=JES` mode, `JOB_OWNER` specifies that commands accessing the JES spool, such as **ls**, and **get**, and so on, apply only to jobs with owner matching the supplied *`name`*.

Default: Current user

`JOBNAME|JESJOB=` *`name`*

When in `FILETYPE=JES` mode, `JOBNAME` specifies that commands accessing the JES spool, such as **ls**, **get**, and so on, apply only to jobs with job name matching the supplied *`name`*.

`KEYLEN|KEYL=` *`length`*

Specifies the length in bytes of the keys used in the data set.

Default: none

`KEYOFF|KEYO=` *`offset`*

Specifies the key offset; the position of the first byte of the key in records of the specified VSAM data set.

Default: none

`LABEL_TYPE|LABEL=NL|SL|NSL|SUL|BLP|LTM|AL|AUL`

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

Note

It is recommended for sites to control the use of `BLP` and `NL` tape processing by restricting access to the appropriate resource, using RACF or an equivalent security product.

`LIKE=` *`like`*

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

You must include the `TYPE` attribute when using `LIKE` unless you are creating a PS data set and the model is a PS data set.

Default: none

`LRECL|R|LR=` *`length`*

Maximum record length or fixed record length.

Default: `4096` for VSAM, `80` if data set organization is `F` or `FB`, otherwise `1024`

`MGMTCLAS|MG=` *`class`*

Specifies the management class of a data set.

Default: none

`NORMDISP|NOR=CATLG|UNCATLG|KEEP|DELETE`

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

Default: `CATLG`

`PRIMARY_SPACE|PRI=` *`space`*

Primary space allocation for a data set.

Default: none

`PROFILE|P|PROF=` *`profile`*

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

Default: none

`RECFM|O|REC=` *`recfm`*

`RECFM` specifies the data set organization. The possible values are all valid combinations of the following letters:

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

Default: `VB`

`RECORD_TRUNCATE|U|TRUN=YES|NO`

When a record truncation occurs while writing an MVS data set, the system will continue writing the data set if `RECORD_TRUNCATE` is set to `YES`; and the system will abort the transfer if `RECORD_TRUNCATE` is set to `NO` or omitted.

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

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

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

The maximum length of a data set record depends on the data set organization:

```
F and FB - LRECL
V and VB - LRECL-4
U        - BLKSIZE
VSAM     - MAXRECLEN
```

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

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

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

Note

When Tectia Client writes a data set with `RECORD_TRUNCATE=YES`, data loss may occur.

`[NO]TRUNCATE|[NO]TRU|[NO]TRUN`

`TRUNCATE|TRU|TRUN` is equal to `RECORD_TRUNCATE=YES`.
`[NO]TRUNCATE|[NO]TRU|[NO]TRUN` is equal to `RECORD_TRUNCATE=NO`.

`RETENTION_PERIOD|RET=` *`days`*

The retention period in days. After the retention period, the data set expires and the operating system can delete or overwrite the data set.

Default: System default

`SECONDARY_SPACE|SE|SEC=` *`space`*

Secondary space allocation for a data set.

Default: none

`SIZE|L=` *`size`*

Size estimate (in bytes) for data set allocation.

Default: `1000000`

`SPACE_RELEASE|RLSE=YES|NO`

When a new data set it allocated, `SPACE_RELEASE` specifies whether unused disk space will be released. If set to `YES`, unused disk space of a new data set is released. If set to `NO`, allocated disk space of a new data set is retained.

Default: `YES`

`SPACE_UNIT|SU=BLKS|TRKS|CYLS|AVGRECLEN`

Unit of space allocation for a data set.

Possible values for the space allocation unit are:

- `BLKS`: Blocks
- `CYLS`: Cylinders
- `TRKS`: Tracks
- `AVGRECLEN`: Average record length

Default: none

`SPACE_UNIT_LENGTH|SUL=` *`length`*

When `SPACE_UNIT=BLKS` or `SPACE_UNIT=AVGRECLEN`, specifies the size of the space allocation unit.

Default: 100 with SPACE_UNIT=AVGRECLEN, none with SPACE_UNIT=BLKS

STAGING|S|STAGE=YES|NO

Specifies whether staging is to be used in the SFTP server when accessing a file or data set.

If set to NO, staging is not used.

If set to YES, staging is used, when needed.

Default: NO

Note

When staging is used, do not set the _CEE_RUNOPTS environment variable's TRAP option to OFF. If you do, **sftpg3** fails to start. The TRAP option is ON by default.

STORCLAS|ST=*class*

Specifies the storage class of system managed storage.

Default: none

SVC99_TEXT_UNITS|SVC99=*string*

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

Default: none

TRACKS|TR

Specifies that the space allocation unit is tracks. Equal to SPACE_UNIT=TRKS.

TRAILING_BLANKS|TRAIL=YES|NO

Specifies whether to preserve trailing blanks in a transferred data set.

If set to YES, trailing blanks will be transferred. This can be used, for example, to preserve the structure of fixed format data sets when transferring to a Unix-type file system.

If set to NO, trailing blanks will be stripped.

Default: NO

Note

This option only applies to line-delimited target files (TRANSFER_FORMAT=LINE), not to target unit-record data sets.

[NO]TRAILINGBLANKS|[NO]TRAI|[NO]TRAIL

TRAILINGBLANKS|TRAI|TRAIL is equal to TRAILING_BLANKS=YES.
NOTRAILINGBLANKS|NOTRAI|NOTRAIL is equal to TRAILING_BLANKS=NO.

`TRANSFER_CODESET|C|CODESET=` *`codeset`*

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

```
> iconv -l
```

Default: none

Example: A Windows SFTP client puts a file to a z/OS data set and gets a data set from z/OS

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

❶ The z/OS server is told that the code set during transfer is ISO8859-1 and that the data set is stored on the server with the IBM-1047 code set.

❷ The server converts the code set from ISO8859-1 to IBM-1047 upon receiving the data.

❸ The server converts the code set from IBM-1047 to ISO8859-1 before sending the data.

Note

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

`TRANSFER_FILE_CODESET|D|FCODESET=` *`codeset`*

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

```
> iconv -l
```

Default: none

`TRANSFER_FILE_LINE_DELIMITER|J|FLDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL`

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

- `UNIX`: The line delimiter used in the file is LF (`\n`, `0x0A`).
- `MVS`: The line delimiter used in the file is NL (`\n`, `0x15`). When writing to a data set, also the CR (`\r`, `0x0D`) code is considered as the End of Line.
- `MVS-FTP`: When reading MVS data sets, each record in the data set is treated as a line. The transfer line delimiter is appended to the record. Any control characters in the record data are preserved.

  When reading data sets with printer control characters, the control characters are preserved in the output.

If the code set conversion is specified either by `TRANSFER_TRANSLATE_TABLE|E`, or by `TRANSFER_CODESET|C` and `TRANSFER_FILE_CODESET|D`, the appended delimiter is the delimiter specified by `TRANSFER_LINE_DELIMITER|I`, `TRANSFER_CODESET|C`, or `TRANSFER_TRANSLATE_TABLE|E`. If no code set conversion is requested, the delimiter is defined by the code set of the data set. By default it is EBCDIC.

You can specify code sets by defining `TRANSFER_FILE_CODESET` without `TRANSFER_CODESET`. For example, to have a DOS delimiter in Unicode (`x'000D000A'`) appended to the records, set `"I=DOS,J=MVS-FTP,D=UCS-2"`, and to have a Unix delimiter in ISO Latin 1 (`x'0A'`), set `"I=UNIX,J=MVS-FTP,D=ISO8859-1"`.

Do not use this when writing data sets.

- `DOS`: The line delimiter used in the file is CRLF (`\r\n`, `0x0D 0x0A`).
- `MAC`: The line delimiter used in the file is CR (`\r`, `0x0D`).
- `NEL`: The line delimiter used in the file is Unicode New Line (`0x85`).

Default: none

Note

The line delimiter information should be given to the host that is capable of performing the conversion, such as a host with a Tectia.

Line delimiter conversion is implemented for single byte code sets only.

For the line delimiter conversion to happen, both `TRANSFER_LINE_DELIMITER|I` and `TRANSFER_FILE_LINE_DELIMITER|J` must be specified.

Example: a z/OS Tectia SFTP client sends a data set to a Windows host and copies the file back from Windows

In this example, the code set is also converted.

```
sftp> lsite I=dos J=mvs                ❶
sftp> lsite C=IBM-437 D=IBM-1047       ❷
sftp> sput //DATASET.TXT file.txt      ❸
sftp> sget file.txt //DATASET.COPY.TXT ❹
```

❶ Transfer line delimiter is set to `DOS` and transfer file line delimiter to `MVS`.

❷ Transfer code set is set to IBM-437 and transfer file code set to IBM-1047.

❸ The z/OS client inserts a NL (`0x15`) character after each record. The line delimiter conversion converts all NL:s to CRLF (`0x0D 0x0A`) characters, which remain unchanged in the code set conversion.

❹ The CRLF line delimiters are converted to LF characters, which are converted to NL characters in the code set conversion. Each NL character (and CR character, if there are any in the data) causes the current record to be written out and a new record started.

`TRANSFER_FORMAT|F|FORMAT=LINE|STREAM|RECORD`

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

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

- `LINE`: The line transfer format is record-based. It uses delimiter characters to mark the end of a record. The delimiter character may be a Carriage Return (CR) or a Newline (NL). When writing to or reading from data sets with ASA control characters, a Form Feed (FF) is also treated as a delimiter. The table below shows the values of these characters in EBCDIC and ASCII. Data sent to Tectia Client in the line transfer format must be in EBCDIC or must be converted to EBCDIC during the transfer.

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

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

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

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

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

  When transferring line format data to and from MVS files with ASA line printer control characters, Tectia Client will convert between the control characters and line delimiter characters, as described in the IBM z/OS *XL C/C++ Programming Guide*, Chapter *"Using ASA Text Files"*.

  To transfer records without changing the ASA code, use the `STREAM` or `RECORD` transfer format, or define the data set using a DD card and specify `RECFM=FB` or `RECFM=VB`.

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

- `STREAM`: The stream transfer format contains the data bytes of the data set but no structural information. If a data set with a fixed record length is transferred with the stream format and

recreated with the same record length, the record structure will be preserved. Variable length records will not be recreated properly if transferred with the stream format.

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

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

Default: `LINE`.

`TRANSFER_LINE_DELIMITER|I|LDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL`

The transfer line delimiter specifies the newline convention used in the data that is transferred over the connection. Possible values are:

- `UNIX`: The line delimiter on the connection is LF (`\n`, `0x0A`).
- `MVS`: The line delimiter on the connection is NL (`\n`, `0x15`). If the data is converted from EBCDIC to ASCII, the NL becomes a LF (`\n`, `0x0A`).
- `MVS-FTP`: When writing to a data set, only the LF (`\n`, `0x0A`) control codes are considered as an End Of Line. Any CR (`\r`, `0x0D`) codes are preserved as data in the record.

  When writing data sets with ASA printer control characters, the first character on each line is used as the ASA character.

  Do not use this when reading data sets.
- `DOS`: The line delimiter on the connection is CRLF (`\r\n`, `0x0D 0x0A`).
- `MAC`: The line delimiter on the connection is CR (`\r`, `0x0D`).
- `NEL`: The line delimiter used in the file is Unicode New Line (`0x85`).

Default: none

Note

The line delimiter information should be given to the host that is capable of performing the conversion, such as a host with a Tectia.

`TRANSFER_MODE|X|MODE=BIN|TEXT`

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

- `BIN`: Code set and line delimiter conversions are not performed.
- `TEXT`: Code set and line delimiter conversions are performed.

Default: none

Note

If TRANSFER_MODE is not given but both TRANSFER_CODESET and TRANSFER_FILE_CODESET or TRANSFER_TRANSLATE_TABLE are present conversions are performed.

TRANSFER_TRANSLATE_DSN_TEMPLATES|A|XDSNT= *templates*

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

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

Note

The translate table must translate line delimiters into EBCDIC NL characters. See TRANSFER_FORMAT.

Default: none

TRANSFER_TRANSLATE_TABLE|E|XTBL= *table*

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

TYPE|T=PS|PO|PDS|POE|PDSE|GDG|HFS|VSAM|ESDS|KSDS|RRN

Specifies the type of a data set when the data set is created. The available values are:

- PS: The type of the created data set is PS.
- PO|PDS: The type of the created data set should be PDS. Note that in order to create a PDS, you need to specify the DIRECTORY_SIZE parameter. If you do not specify the directory size, a sequential data set - not a partitioned data set - is created.
- POE|PDSE: The type of the created data set is PDSE.
- GDG: The type of the created data set is GDG.
- HFS: The type of the created data set is HFS.
- VSAM: The type of the created data set is VSAM.

- `ESDS`: The type of the created data set is VSAM ESDS.
- `KSDS`: The type of the created data set is VSAM KSDS.
- `RRN`: The type of the created data set is VSAM RRN.

Default: `PO`, if data set name includes member, otherwise `PS`

`UNIT|UN=` *`unit`*

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

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

Default: none

`UNIT_COUNT|UC|UNC=` *`number`*

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

Default: System default

`UNIT_PARALLEL|UNP=YES|NO`

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

Default: System default

`VOLUME_COUNT|VC|VOLCNT=` *`number`*

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

Default: System default

`VOLUMES|VO|VOL=` *`vol1`*`+`*`vol2`*`+...`

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

Default: none

Chapter 6 Secure Shell Tunneling

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

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

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

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

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

6.1 Local Tunnels

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

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

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

where:

- `[protocol/]` specifies which protocol is to be used in the tunneled connection, it can be `ftp` or `tcp` (optional argument). The default is `tcp`.
- `[listen-address:]` defines which interface on the local client will be listened to (optional argument). If it is omitted, only local interface is listened unless the `-g --gateway` option is used before `-L` to bind to all interfaces on the client-side.

- `listen-port` is the number of the port on the local client, and connections coming to this port will be tunneled to the server.
- `dst-host:dst-port` define the destination host address and the port to which the connection is tunneled from the server.
- `sshserver` is the IP address or the host name of the Secure Shell server.

The host name or IP address of the destination host and sshserver can be defined as regular expressions that follow the egrep syntax, but no wildcards are supported.

Note

If `dst-host` is specified as a domain name rather than IP address, the name will be resolved according to the address family settings of `sshserver`. For example, if the domain name is resolved to an AAAA DNS record (IPv6) and the address family setting of the server is `inet` (IPV4), the tunnel will not work.

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

Note

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

Figure 6.1 shows the different hosts and ports involved in local tunneling (port forwarding).

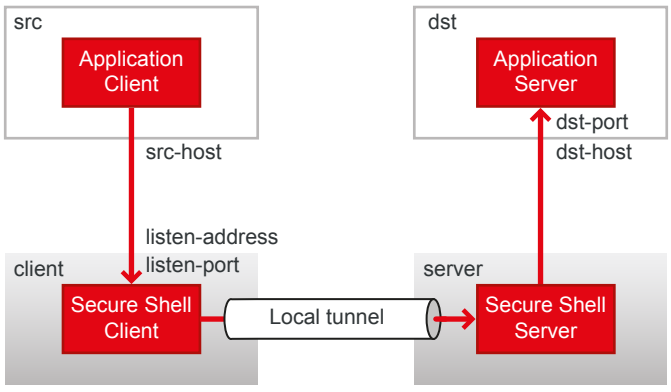


Figure 6.1. Local tunneling terminology

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

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

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

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

6.1.1 Transparent TCP Tunneling on Windows

You can configure Tectia Client on Windows to automatically tunnel outbound TCP connections via secure SSH tunnels.

Note

Using this feature requires that you've installed Tectia Client's **Transparent TCP Tunneling** module. If needed, you can install this by running the Tectia Client installation wizard.

A Tectia Broker filters outbound TCP connections and creates secure SSH tunnels between the Broker and the SSH Server. You can configure Tectia Broker to tunnel, block, or permit connections. You can configure filtering per application name, remote IPv4/IPv6 address, hostname, and port.

There are two brokers for filtering connections:

- **Trusted broker**: Any filter rules configured in the trusted broker have priority over user-broker rules. By default does not do any filtering of its own, only helps user brokers with hostname filtering. Runs as SYSTEM by default.
- **User broker**: Broker with user-specific filtering rules for outbound TCP and DNS.

If you only want to filter using one type of broker, you may disable either broker type.

Transparent tunneling on Windows also relies on the `capture driver`. The capture driver captures the outbound TCP connections and DNS queries and asks first the trusted Broker (if configured) and then the user specific Broker (if configured) what to do with these TCP connections and DNS queries.

Configuring Transparent Tunneling

Rules for transparent tunneling are configured in the brokers' `filter-engine` element. For more information about the element's syntax, see `filter-engine`.

To configure the trusted broker, run the following with Administrator privileges. Or if you have a non-default trusted user, as the trusted user:

```
ssh-tectia-configuration.exe -a capture-trusted-broker
```

The host keys of transparent-tunnel targets must be in the known hosts file. Otherwise the tunneling will fail. You can add a host key to a user broker with:

```
$ ssh-broker-ctl probe-key --save-hostkey ip#port
```

To save a host key to the trusted broker, run the following with Administrator privileges. Or if you have a non-default trusted user, as the trusted user:

```
# ssh-broker-ctl -a capture-trusted-broker probe-key --save-hostkey ip#port
```

If you want to use the trusted broker as a regular user, set your broker with:

```
$ set SSH_SECSH_BROKER=capture-trusted-broker
```

Any subsequent client commands (sshg3, scpg3, sftpg3) shall use the trusted broker for filtering connections.

In the global broker configuration file, the `windows-capture` section allows you to configure trusted-broker settings, such as trusted user, filtering, automatic start on boot etc. After changes to the `windows-capture` section, you will need to run the following:

```
# ssh-broker-ctl capture-driver reconfig
```

The capture driver is managed via **ssh-broker-ctl capture-driver**. For more information about available commands and options, such as capture-driver setup and debug logging, see the section called "Commands"

Transparent Tunneling Logs

You can obtain information about transparent-tunneling components for logging and troubleshooting purposes.

For information about the Tectia Broker in transparent tunneling, run:

```
$ ssh-broker-ctl status
```

To view the status of the trusted broker, run the following as Administrator when using a default trusted user, or as the trusted user when using a custom trusted user:

```
# ssh-broker-ctl -a capture-trusted-broker status
```

You can review capture driver logs from Windows Event Logs:

- For major events, see **Applications and Services Logs→SSHCaptureDriver→Operational**
- For debug logs, go to **Applications and Services Logs→SSHCaptureDriver**, right click **SSHCaptureDriver** and select **View→Show Analytic and Debug Logs**, then open **SSHCaptureDriver→Debug**.

Note

Debug logs are disabled by default. They can be enabled with **ssh-broker-ctl capture-driver debug <*lvl*>**. To enable debug logs for the trusted broker, run the command as an Administrator/custom trusted user with the added option **-a capture-trusted-broker**.

We do not recommend leaving debug logging enabled for prolonged time, at least not with levels 4 and greater. Doing so may result in lots of logs and consume memory.

6.1.2 Non-Transparent TCP Tunneling

When non-transparent TCP tunneling is used, the application to be tunneled is set to connect to the local listener port instead of connecting to the server directly. Tectia Client forwards the connection securely to the remote server.

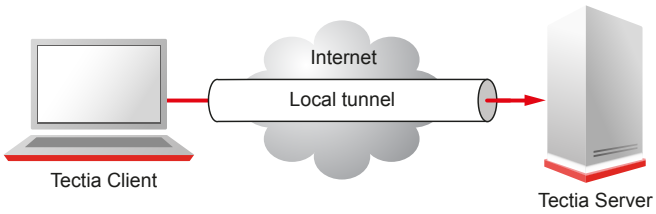


Figure 6.2. Simple local tunnel

If you have three hosts, for example, `sshclient`, `sshserver`, and `imapserver`, and you forward the traffic coming to the `sshclient`'s port `143` to the `imapserver`'s port `143`, only the connection between the `sshclient` and `sshserver` will be secured. The command you use would be similar to the following one:

```
sshclient$ sshg3 -L 143:imapserver:143 username@sshserver
```

Figure 6.3 shows an example where the Secure Shell server resides in the DMZ network. Connection is encrypted from the Secure Shell client to the Secure Shell server and continues unencrypted in the corporate network to the IMAP server.

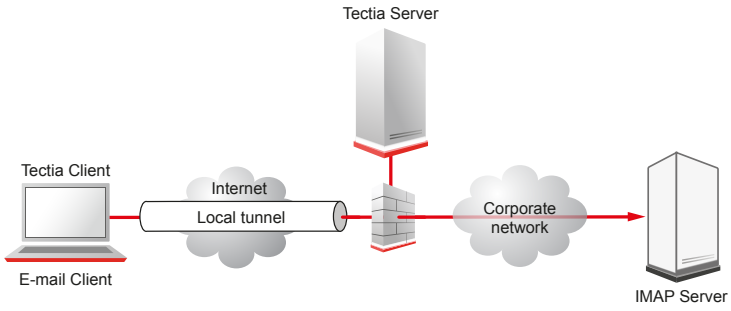


Figure 6.3. Local tunnel to an IMAP server

Tunnels can also be defined for connection profiles in the Connection Broker configuration file. The defined tunnels are opened automatically when a connection with the profile is made. The following is an example from a `ssh-broker-config.xml` file:

```
<profile id="id1" host="sshserver.example.com">
...
  <tunnels>
    <local-tunnel type="tcp"
                  listen-port="143"
                  dst-host="imap.example.com"
                  dst-port="143"
```

```
                  allow-relay="no" />
  ...
  </tunnels>
</profile>
```

By default, local tunnels originating only from the client host itself are allowed. To allow also other machines to connect to the tunnel listener port, set the `allow-relay` to `yes`.

The tunneling settings can be made in the Tectia Connections Configuration GUI, under **Connection Profiles → Tunneling** per each profile. See the section called “Defining Tunneling”.

Automatic Tunnels

Automatic tunnels are one way of creating non-transparent local tunnels for application connections.

Automatic tunnels always use a connection profile in the tunnel establishing. You can create listeners for local tunnels that will be activated automatically when the Connection Broker starts up. The actual tunnel will be formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.

In the Connection Broker configuration file, make the following kind of settings:

```
<static-tunnels>
  <tunnel type="tcp"
          listen-port="9874"
          dst-host="st.example.com"
          dst-port="9111"
          allow-relay = "no"
          profile="id1" />
</static-tunnels>
```

You can configure the automatic tunnels in the Tectia Connections Configuration GUI, on the **Automatic Tunnels** page. For instructions, see Section A.1.6.

Examples of Local Tunneling

When **sshg3** is used to create secure tunnels using local port forwarding, the TCP applications to be tunneled are configured to connect to a localhost port instead of the application server port.

Example application, `clientapp1`, by default connects to a Unix server `unix.example.com` using TCP port 2345.

```
$ clientapp1 --username user1 --server unix.example.com --port 2345
```

For securing this TCP application using Secure Shell, use the following commands:

```
$ sshg3 -L 2345:localhost:2345 user1@unix.example.com -S -f &
$ clientapp1 --username user1 --server localhost --port 2345
```

The above **sshg3** command connects to remote Secure Shell server `unix.example.com`, creates a local listener on port 2345, instructs the remote Secure Shell server to forward the incoming traffic to `localhost:2345`, and goes to background in single-shot-mode.

6.1.3 Non-Transparent FTP Tunneling

Non-transparent FTP tunneling is an extension to the generic tunneling mechanism. Unlike generic tunneling (port forwarding) mechanism, non-transparent FTP tunneling secures the transferred files, in addition to the FTP control channel. The FTP tunneling code monitors the tunneled FTP control channels and dynamically creates new tunnels for the data channels as they are requested.

When non-transparent FTP tunneling is used, tunnels are created from local client ports to remote servers. The FTP client is configured to connect to Tectia Client which will forward the connection to the endpoint where a Secure Shell server is running.

The typical use case is that Tectia Client is located on the same host as the FTP client; and the FTP server is on the same host as the Secure Shell server. However, other configurations are also supported, but it is worth noticing that the connection is encrypted only between Tectia Client and the Secure Shell server.

Non-transparent FTP tunneling can be requested on the command line, or enabled and defined in the Connection Broker configuration. The configured non-transparent FTP tunneling uses connection profiles, that are defined on Tectia Client.

On command-line, FTP tunneling can be used for both local and remote tunnels. Non-transparent FTP-tunneling is started by entering a `sshg3` command with the following syntax:

```
sshclient$ sshg3 -L ftp/1234:localhost:21 username@sshserver
```

For information on the `sshg3` command, see the sshg3(1) man page.

The FTP tunneling settings can be made in the Tectia Connections Configuration GUI, under **Connection Profiles → Tunneling** for each profile. See the section called “Defining Tunneling”.

FTP tunnels can also be defined for connection profiles in the Connection Broker configuration file. The following is an example from the Connection Broker configuration file `ssh-broker-config.xml`:

```
<profiles>
  <profile id="id1" host="sshserver.example.com"
  ...
    <tunnels>
      <local-tunnel type="FTP"
                    listen-port="1234"
                    dst-host="127.0.0.1"
                    dst-port="21"
                    allow-relay="NO" />
      ...
    </tunnels>
  </profile>
</profiles>
```

An FTP connection can then be made with the following (example) commands:

```
sshclient$ ftp
ftp$ open localhost 1234
```

The FTP connection to port 1234 on client is now tunneled to port 21 on the Secure Shell server.

As an alternative to FTP tunneling, you can use the **sftpg3** or **scpg3** clients for secure file transfers. These clients can be used on command line or in scripts and they require less configuration than FTP tunneling, since Tectia Server already has **sft-server-g3** as a subsystem, and **sftpg3** and **scpg3** clients are included with Tectia Client. Managing remote user restrictions on the server machine will be easier, since you do not have to do it also for FTP.

To understand exactly how FTP tunneling is done, two different cases need to be examined: the active mode and the passive mode of the FTP protocol.

Tunneling FTP in Passive Mode

In passive mode, the FTP client sends the command `PASV` to the server, which reacts by opening a listener port for the data channel and sending the IP address and port number of the listener as a reply to the client. The reply is of the form

```
227 Entering Passive Mode (a1,a2,a3,a4,p1,p2)
```

where `a1.a2.a3.a4` is the IP address and `p1*256+p2` is the port number. For example, the reply for IP address 10.1.60.99 and port 1548 is: `227 Entering Passive Mode (10,1,60,99,6,12)`.

When the Connection Broker notices the reply to the `PASV` command, it will create a local port forwarding to the destination mentioned in the reply. After this the Connection Broker will rewrite the IP address and port in the reply to point to the listener of the newly created local port forwarding (which exists always in a localhost address, 127.0.0.1) and pass the reply to the FTP client. The FTP client will open a data channel based on the reply, effectively tunneling the data through the Secure Shell connection, to the listener the FTP server has opened. The net effect is that the data channel is secured all the way except from the Secure Shell server to the FTP server if they are on different machines. This sequence of events takes place automatically for every data channel.

Since the tunnel is opened to a localhost address, the FTP client must run on the same machine as Tectia Client if passive mode is used.

Tunneling FTP in Active Mode

In active mode, the FTP client creates a listener on a local port, for a data channel from the FTP server to the FTP client, and requests the channel by sending the IP address and the port number to the FTP server in a command of the following form:

```
PORT a1,a2,a3,a4,p1,p2
```

where `a1.a2.a3.a4` is the IP address and `p1*256+p2` is the port number. The Connection Broker intercepts this command and creates a remote port forwarding from the localhost address of the Secure Shell server to the address and port specified in the `PORT` command.

After creating the tunnel, the Connection Broker rewrites the address and port in the `PORT` command to point to the newly opened remote forwarding on the Secure Shell server and sends it to the FTP server. Now the FTP server will open a data channel to the address and port in the `PORT` command, effectively forwarding the data through the Secure Shell connection. The Connection Broker passes the incoming data to the original listener created by the FTP client. The net effect is that the data channel is secure the

whole way except from Tectia Client to the FTP client. This sequence of events takes place automatically for every data channel.

For FTP tunneling in active mode to work, the FTP server must be run on the same host as the Secure Shell server, and the FTP client and Tectia Client must reside on the same host.

Note

Tunneling FTP in active mode is not guaranteed to work in all setups. If possible, use the passive mode when tunneling FTP connections.

6.1.4 SOCKS Tunneling

SOCKS tunneling is a mechanism available for tunneling applications that support the SOCKS4 or SOCKS5 client protocol.

Instead of configuring tunneling (a.k.a port forwarding) from specific ports on the local host to specific ports on the remote server, you can specify a SOCKS server which can be used by the user's applications. Each application is configured in the regular way except that it is configured to use a SOCKS server on a localhost port. The Secure Shell client application, Tectia Client, opens a port in the localhost and mimics a SOCKS4 and SOCKS5 server for any SOCKS client applications.

When the applications connect to services such as IMAP4, POP3, SMTP, and HTTP, they provide the necessary information to the SOCKS server, which is actually Tectia Client mimicking a SOCKS server. Tectia Client will use this information in creating a tunnel to the Secure Shell server and relaying the traffic back and forth securely.

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

```
client$ sshg3 -L socks/[listen-address:]listen-port username@sshserver
```

where:

- `[listen-address:]` defines which interface on the client will be listened to (optional argument)
- `listen-port` is the number of the port on the client
- `sshserver` is the IP address or the host name of the Secure Shell server.

For example, the following command will set up a local tunnel from port `1234` on the client to `sshserver`. The applications are set to use a SOCKS server at port `1234` on the client. From the server, the connections are forwarded unsecured to the destination hosts requested by the applications.

```
sshclient$ sshg3 -L socks/1234 username@sshserver
```

SOCKS tunnels can also be defined for connection profiles in the Connection Broker configuration file. The following is an example from a `ssh-broker-config.xml` file:

```
<profile id="id1" host="sshserver.example.com">
...
  <tunnels>
    <local-tunnel type="socks"
```

```
                listen-port="1234"
                allow-relay="no" />
  ...
  </tunnels>
</profile>
```

6.2 Remote Tunnels

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

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

```
client$ sshg3 -R [protocol/][listen-address:]listen-port:dst-host:dst-port \
username@sshserver
```

where:

- `[protocol/]` specifies which protocol is to be used in the tunneled connection, it can be `ftp` or `tcp` (optional argument). The default is `tcp`.
- `[listen-address:]` defines which interface on the remote server will be listened to (optional argument). If it is omitted, only local interface is listened unless the `-g --gateway` option is used before `-R` to bind to all interfaces on the server-side.
- `listen-port` is the number of the port on the remote server, and connections coming to this port will be tunneled to the client.
- `dst-host:dst-port` define the destination host address and the port to which the connection is tunneled from the client.
- `sshserver` is the IP address or the host name of the Secure Shell server.

The IP addresses and host names of the destination host and the sshserver can be defined using regular expressions that follow the egrep syntax. No wildcards are supported.

Note

If `dst-host` is specified as a domain name rather than IP address, the name will be resolved according to the address family settings of `client`. For example, if the domain name is resolved to an AAAA DNS record (IPv6) and the address family setting of the client is `inet` (IPV4), the tunnel will not work.

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

Note

Every user with access to the remote server host will be able to use remote tunnel.

Figure 6.4 shows the different hosts and ports involved in remote port forwarding.

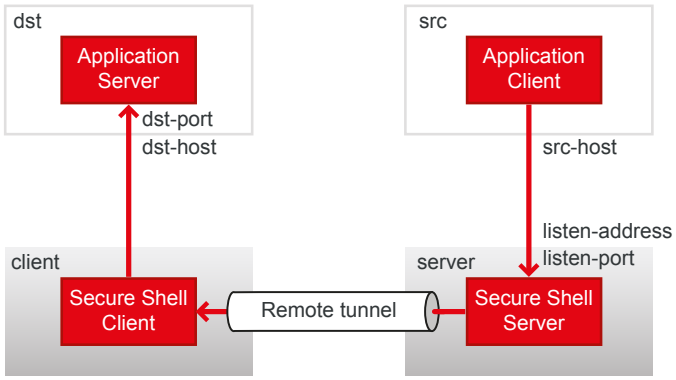


Figure 6.4. Remote tunneling terminology

For example, if you issue the following command, all traffic which comes to port 1234 on the server will be tunneled to port 23 on the client. See Figure 6.5.

```
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.

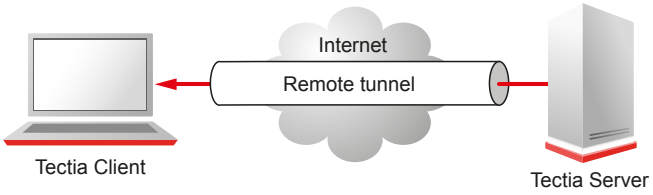


Figure 6.5. Remote tunnel

Tunnels can also be defined for connection profiles in the Connection Broker configuration file. The defined tunnels are opened automatically when a connection with the profile is made.

The following is an example from a `ssh-broker-config.xml` file:

```
<profile id="id1" host="sshserver.example.com">
 ...
   <tunnels>
      <remote-tunnel type="tcp"
                     listen-port="1234"
                     dst-host="localhost"
                     dst-port="23" />
   ...
   </tunnels>
</profile>
```

The tunneling settings can be made in the Tectia Connections Configuration GUI, under **Connection Profiles → Tunneling** per each profile. See the section called “Defining Tunneling”.

6.3 X11 Forwarding

X11 forwarding is a special case of remote tunneling.

Tectia Client supports X11 forwarding on both Unix and Windows platforms. On Windows, you need also the XWindow Manager package. Tectia Server supports X11 forwarding only on Unix platforms.

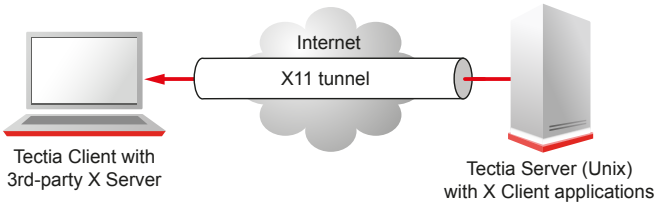


Figure 6.6. X11 forwarding

X11 forwarding can be enabled in the client by setting the following line in the `ssh-broker-config.xml` file (either under `default-settings` or under a connection `profile`):

```
<forwards>
  <forward type="X11" state="on"/>
</forwards>
```

By default, X11 forwarding is off.

X11 forwarding can be enabled in the Tectia Connections Configuration GUI, under **Default Connection → Tunneling** for the default connection, and under **Connection Profiles → Tunneling** per each profile. See the section called "Defining Default Tunneling Settings" and the section called "Defining Tunneling".

To test that X11 forwarding works on Windows, use the XWindow Manager. Log into the remote system and type `xclock &`. This starts an X clock program that can be used for testing the forwarding connection.

If the X clock window is displayed properly, you have the X11 forwarding working. If the X clock fails and complains that it cannot open the display, check that the XAuth is properly installed on the remote host.

Note

Do *not* set the `DISPLAY` variable on the client. You will most likely disable encryption. (X connections tunneled through Secure Shell use a special local display setting.)

6.4 Agent Forwarding

Agent forwarding is a special case of remote tunneling. In agent forwarding, Secure Shell connections and public-key authentication data are forwarded from one server to another without the user having to authenticate separately for each server. Authentication data does not have to be stored on any other machine than the local machine, and authentication passphrases or private keys never go over the network.

Tectia Client provides 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.

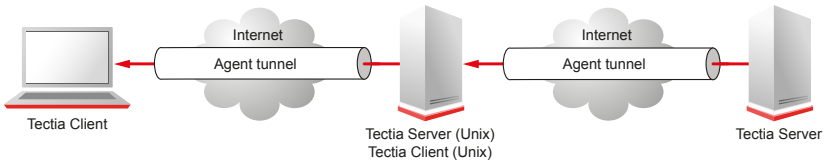


Figure 6.7. Agent forwarding

In the factory settings, agent forwarding is enabled (on).

Agent forwarding can be enabled or disabled on the client side both in the default configuration settings and separately for each connection profile.

In the `ssh-broker-config.xml` file, agent forwarding can be disabled by setting the following line either under `default-settings` or under a connection `profile`:

```
<forwards>
  <forward type="agent" state="off" />
</forwards>
```

Agent forwarding can be disabled in the Tectia Connections Configuration GUI, under **Default Connection → Tunneling** for the default connection, and under **Connection Profiles → Tunneling** per each profile. See the section called "Defining Default Tunneling Settings" and the section called "Defining Tunneling".

Chapter 7 Troubleshooting Tectia Client

If you encounter any connection, authentication, or configuration problems, you can try to solve them by running Tectia Client in debug mode. For more information on debugging, see instructions in Section 7.3.

You can also gather information needed for troubleshooting and send it to SSH technical support. See Section 1.2 for information on accessing the Tectia online support resources and contacting SSH technical support.

The information you need to provide to SSH technical support includes:

- The verbose level output of Tectia Client; see Section 7.1.
- System information from Tectia Troubleshooting Tool's output. This information will help in analyzing the reported problems, as the technical support gets to know the exact details about the environment where the Tectia products are running; see Section 7.2.
- Possibly debug information; see Section 7.3.

7.1 Gathering Basic Troubleshooting Information

Most connection problems can be solved by running **sshg3** in verbose mode and examining the output.

Enter the command **-v** (or **--verbose**) to get the diagnostic output:

```
$ sshg3 -v user@server.example.com
```

You can also get the diagnostic output from the old connection attempts afterwards. The following command lists the old connection attempts and their connection IDs:

```
$ ssh-broker-ctl list-connections --disconnected
```

The following command shows the diagnostic output from the old connection attempts:

```
$ ssh-broker-ctl connection-status <connection-id>
```

7.2 Collecting System Information for Troubleshooting

Tectia Client 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.

7.3 Setting Connection Broker to Debug Mode

The Connection Broker is a component included in Tectia Client. The Connection Broker handles all cryptographic operations and authentication-related tasks for Tectia Client and the command-line tools **sshg3, scpg3**, and **sftpg3**.

If the verbose level output explained in Section 7.1 does not solve your problem, set the existing running Broker to debug mode. Existing open connections will remain up and running, which is relevant on multiuser systems or when there are lots of automated scripts running at the same time. You will also get a debug log from new connection attempts.

To set the Connection Broker to debug mode, follow these instructions:

1. Open a shell (on Unix) or command prompt window (on Windows).
2. If you already have an existing Connection Broker, skip this step. If you do not have an existing Connection Broker, run the following command:

   ```
   $ ssh-broker-g3
   ```

3. Set the Connection Broker to debug mode by running the following command:

   ```
   $ ssh-broker-ctl debug --log-file=<logfile> <debug-level>
   ```

   In the command:

   - `logfile` specifies the file to which the debug output will be directed
   - `debug-level` is an integer from 0 (no debug info) to 99 that specifies the desired amount of debug information.

   ### Note

   The recommended debug levels are 1-9. The higher the number, the more detailed the troubleshooting output will be, and the more the debugging will affect performance.

   You can set the debug mode also in the **Debug Log** tab in the **PrivX Desktop**. To open the **Debug Log** tab, click the top-right **menu** and select **View Debug Log**.

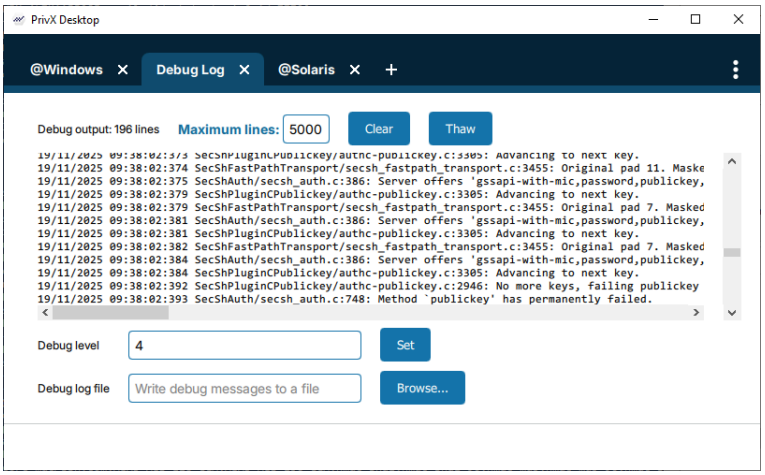


Figure 7.1. Setting the Connection Broker's debug mode with GUI

The following example command sets the Connection Broker debug mode to level 4 and outputs the debug information to a log file named `broker.log`:

```
$ ssh-broker-ctl debug --log-file=broker.log 4
```

4. Connect to a server using one of the clients:

```
$ sshg3 user@host
```

5. View the debug information for the connection in the `broker.log` file.

You can display the debug output also by using the command line tools with argument `-D`. For example, the following command will display the debug output with a debug level 4 and set the same debug for Connection Broker as well:

```
$ sftpg3 -D '4;broker' user@host
```

On Windows, besides the command line tools, you can display the debug output also in the **Tectia Connection Status** window.

Note

After you have collected the debug output, remember to disable Tectia Client's debug mode, since debugging slows down the performance.

On Unix and Windows, the debug mode is disabled with the following command:

```
$ ssh-broker-ctl debug --clear
```

On Windows, the debug mode can be also disabled by setting the debug level back to 0 in the **Debug Log** tab, as shown in Figure 7.2

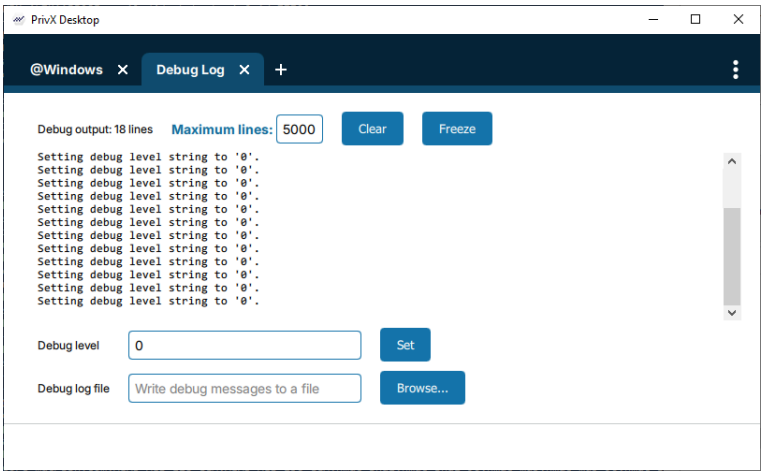


Figure 7.2. Disabling the Connection Broker's debug mode in GUI

7.4 Answers to Common Problems

This sections introduces workaround instructions for some problem situations.

Troubleshooting GSSAPI Authentication

When connecting from a Windows 5.x or 6.x client to a Windows 4.x server using GSSAPI authentication, if authentication fails although GSSAPI has been correctly configured, you may have to disable the `LMHOSTS` lookup on the client-side computer. Follow these instructions:

1. Select **Control Panel → Network Connections**.
2. In **Local Area Connection**, right-click and select **Properties**.
3. In the **Local Area Connection Properties** dialog box, **General** tab, select **Internet Protocol (TCP/IP)** and click the **Properties** button.
4. In the **Internet Protocol (TCP/IP) Properties** dialog box, in the **General** tab, click the **Advanced** button.
5. In the **Advanced TCP/IP Settings** dialog box, in the **WINS** tab, clear the **Enable LMHOSTS lookup** check box.
6. Restart the client-side computer.

Publickey authentication fails

If there are multiple public-keys available either via external key providers or in the user-specific configuration directory, the server-side might refuse the connection if too many keys are attempted during publickey authentication. Typically the error shown when attempting to connect to an OpenSSH server that allows six attempts is "Connection open failed Protocol error (remote): too many authentication failures public_keys_tried: 7". If the server configuration cannot be changed to

allow more attempts, the workaround on the client-side is to set the Connection profile to prompt the user for the specific key to use or with the sshg3, sftpg3 or scpg3 command-line clients use the -K option to specify the explicit key.

```
sshg3 -K <path_to_key> <profilename>
```

Appendix A Connection Broker Configuration Tools

The Connection Broker is a component included in Tectia Client, and it handles all cryptographic operations and authentication-related tasks for Tectia Client. For this reason, all authentication and connection profile settings are made in the Connection Broker configuration.

The Connection Broker configuration can be edited and viewed via the Tectia Connections Configuration GUI. For instructions, see Section A.1.

The Connection Broker stores the settings in an XML-based configuration file `ssh-broker-config.xml`. It is possible to edit the configuration file directly with an ASCII-text editor or an XML editor. For a detailed description of the configuration file, see ssh-broker-config(5). For a quick reference to the configuration file's elements and their attributes, see Section A.4.

A.1 Tectia Connections Configuration GUI

You can use the Tectia Connections Configuration GUI to edit the authentication and connection profile settings on the Connection Broker included in Tectia Client.

Tectia Connections Configuration GUI is available on Windows, Linux and macOS for Tectia Client. There are some differences in the GUI options between different OS platforms and product versions. The following screen shots typically show Tectia Client on Windows. When the differences are important, the differences are noted.

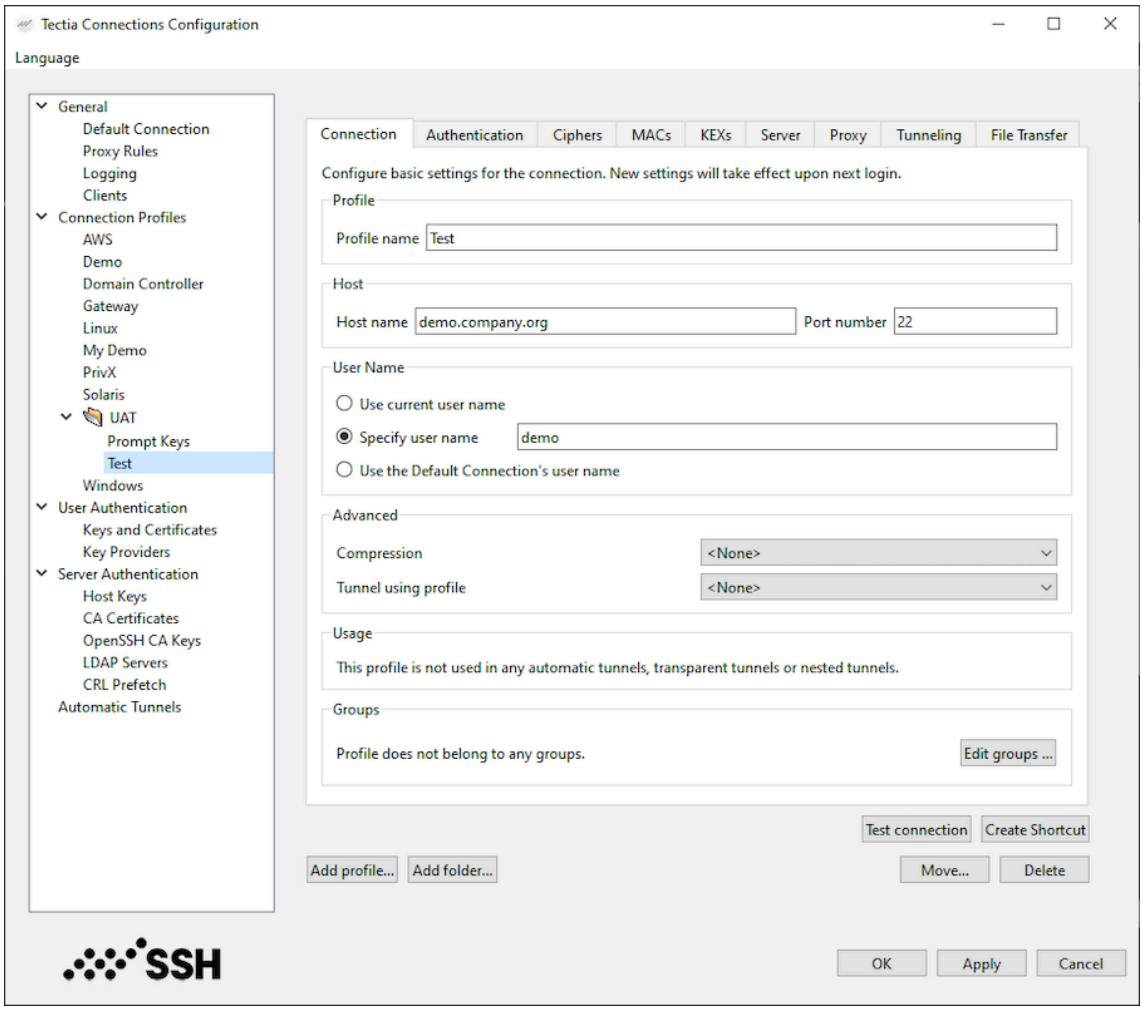


Figure A.1. Connection profile tabs

On Linux and macOS, the Tectia Connections Configuration GUI interface differs from the Windows version most notably with the following exceptions:

- The `Microsoft Crypto API` option on the **Key Providers** page is not available.

Note

Only KDE and Gnome window managers are supported. For other compatible managers, check the website for the Qt framework.

A.1.1 Opening the GUI

When PrivX Desktop GUI is running

The **Tectia Connections Configuration GUI** can be accessed in several ways:

- When you have the PrivX Desktop GUI active, click the menu icon and select **Configuration** or right-click on a specific profile tile and select **Edit profile...** on **Profiles** page.

- On Windows, right-click on the PrivX Desktop tray icon  in the Windows taskbar notification area to access the shortcut menu, and select **Configuration**.
- On Linux and macOS, click on the PrivX Desktop tray icon  in menu bar to access the tray icon menu, and select **Configuration**.

On command-line

The **Tectia Connections Configuration GUI** can be started by running on command-line:

```
ssh-tectia-configuration
```

The following options are available:

`-f, --config=`*`FILE`*

Use configuration file *`FILE`*.

`-a, --broker-address=`*`ADDR`*

Connect to separate Connection Broker process using given address.

`-d, --debug=`*`STR`*

Sets debug string to *`STR`*.

`--convert`

Convert old configuration file.

`--new-profile`

Add a new profile.

`--profile-host=`*`HOST`*

Profile host name when adding a new profile.

`--profile-port=`*`PORT`*

Profile port when adding a new profile.

`--profile-user=`*`USER`*

Profile user when adding a new profile.

`--edit-profile=`*`NAME`*

Edit existing profile *`NAME`*.

`--ui-mode=`*`MODE`*

User interface mode. Possible values are `standard` and `file-transfer`.

`-V, --version`

Print version.

`-h, --help`

Print usage.

A.1.2 Defining General Settings

On the **General** page, you can select the cryptographic library to be used and define the Tectia tray icon settings.

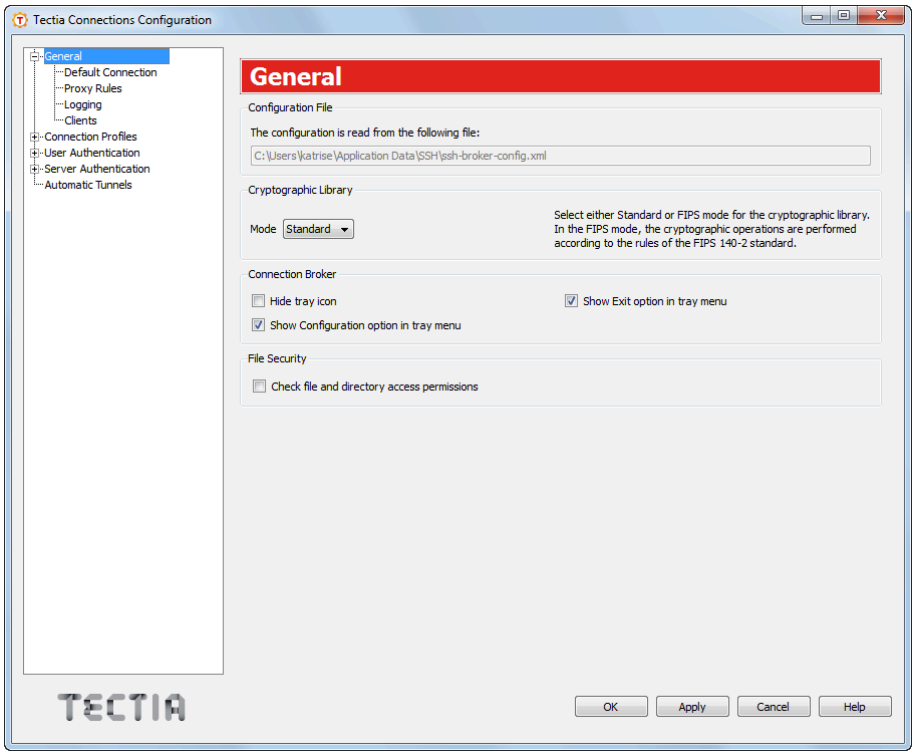


Figure A.2. General settings

Configuration File

Shows the location of the user-specific Broker configuration file. The default location is "`%APPDATA%\SSH\ssh-broker-config.xml`" on Windows and "`$HOME/.ssh2/ssh-broker-config.xml`" on Linux.

Each time the configuration file is saved, a backup of the old configuration is stored in "`%APPDATA%\SSH\ssh-broker-config.xml.bak`" on Windows and "`$HOME/.ssh2/ssh-broker-config.xml.bak`" on Linux.

Cryptographic Library

Tectia Client can be operated in *FIPS mode*, using a version of the cryptographic library that has been validated according to the Federal Information Processing Standard (FIPS) 140-2. In this mode,

the cryptographic operations are performed according to the rules of the FIPS 140-2 standard. The OpenSSL cryptographic library is used in the *FIPS mode*.

Select whether to use the **Standard** or the **FIPS** 140-2 certified version of the cryptographic library.

For the default settings, see the section called “Defining Ciphers”, the section called “Defining MACs”, and for the profile-specific settings, see the section called “Defining Ciphers”, and the section called “Defining MACs”.

Connection Broker

Select whether to hide the Tectia tray icon from the Windows taskbar notification area, and whether to show the **Exit** and **Configuration** options in the shortcut menu.

File Security *(Available on Linux, only)*

Select the **Check file and directory access permissions** check box to enable checking the access permissions for the user-specific configuration file (`$HOME/.ssh2/ssh-broker-config.xml`) and the private key files. By default, file and directory access permissions are not checked.

When the file and directory access permissions are checked, the controls are applied as follows:

- Expected permissions for the user configuration file: only the user has read and write rights. If the permissions are any wider, the Connection Broker will not start.
- Expected permissions for the private key files: only the user has read and write rights. If the permissions are any wider, keys that do not pass the check will be ignored.

Defining Default Connection Settings

The **Default Connection** page allows you to edit the default settings for a user name (the section called “Defining Connection Settings”), authentication (the section called “Defining Authentication”), ciphers (the section called “Defining Ciphers”), MACs (the section called “Defining MACs”), KEXs the section called “Defining KEXs”, server connections (the section called “Defining Server Connections”), and tunneling (the section called “Defining Default Tunneling Settings”).

Newly created connection profiles will inherit the default settings defined here. The values can be customized on the profile-specific tabbed pages and they override the default settings. See the section called “Defining Authentication”, the section called “Defining Ciphers”, the section called “Defining MACs”, and the section called “Defining Server Connections”.

Defining Connection Settings

On the **Connection** tab, you can define a default user name to be used when connecting to remote servers. This connection is useful when several users will be using profiles jointly, either with their own system user names or with a common user account.

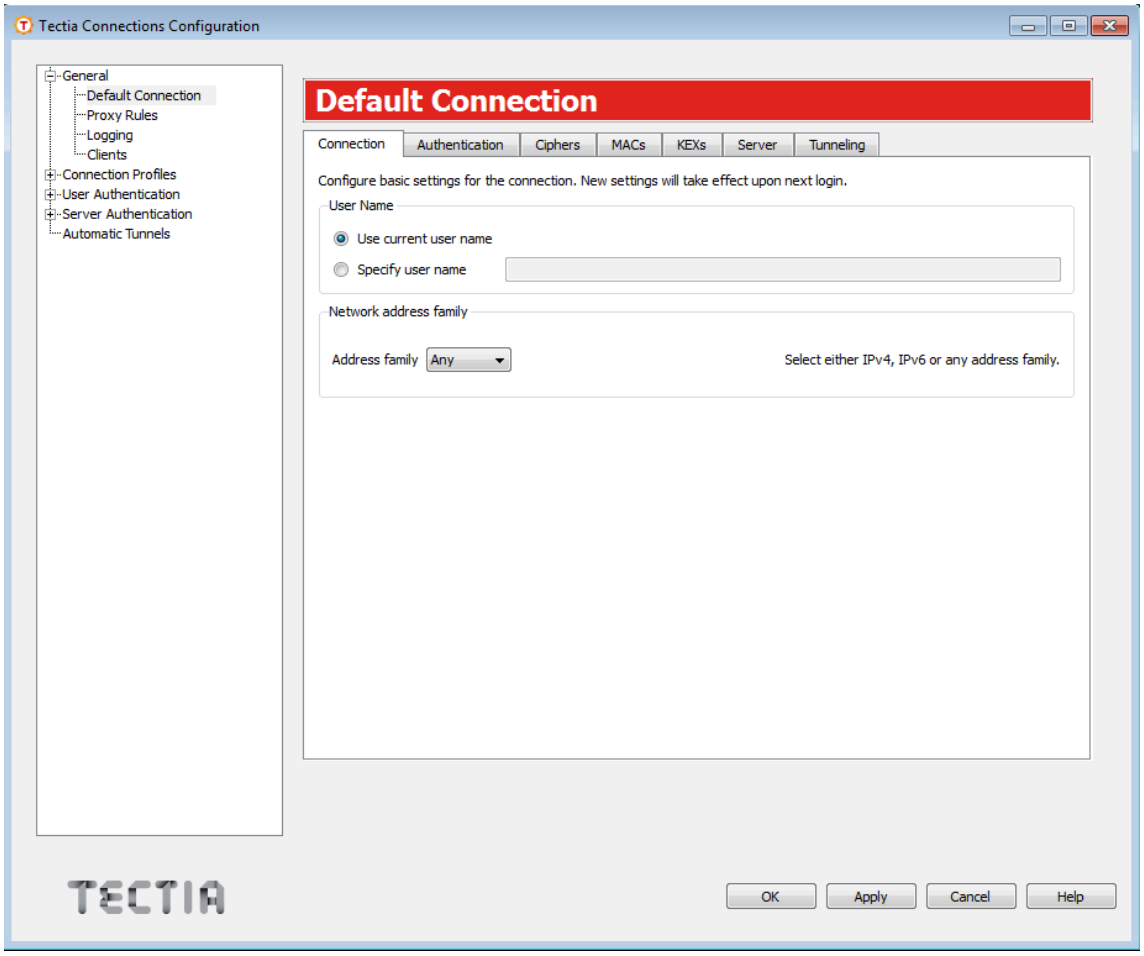


Figure A.3. The user name and network address family settings for connections

Select the **Use current Windows user name** option, to automatically apply the Windows user name of the currently logged in user to connections to remote servers.

Select the **Specify user name** option and enter a generic user name. Note that the name is case sensitive.

The given user name will be used in connections unless another user name is specified in a connection profile or connection attempt. In case you select this option but leave the user name field empty, the Connection Broker will prompt the user for a user name.

In principle, you can enter value "`%USERNAME%`", but it has the same effect as selecting **Use current Windows user name**.

If you specify a host name or the profile contains a host name, the Connection Broker will try to resolve the address based on the **Network address family** setting. If you select `inet`, the Connection Broker will resolve the host name only with an IPv4 address. If you select `inet6`, the Connection Broker will resolve the host name only with an IPv6 address. Selecting `Any` means that the Connection Broker will resolve the host name with any IP address (IPv4 or IPv6) available.

Note

You can specify a direct IP address (either IPv4 or IPv6) for the connection using either the connection profile or the command line. This setting does not restrict the user specified network family address. For example, the connection will be established to a specified IPv4 address even if the network address family was set to IPv6.

Settings made in this tab take effect the next time a user logs in.

Defining Authentication

On the **Authentication** tab, you can define the default user authentication methods.

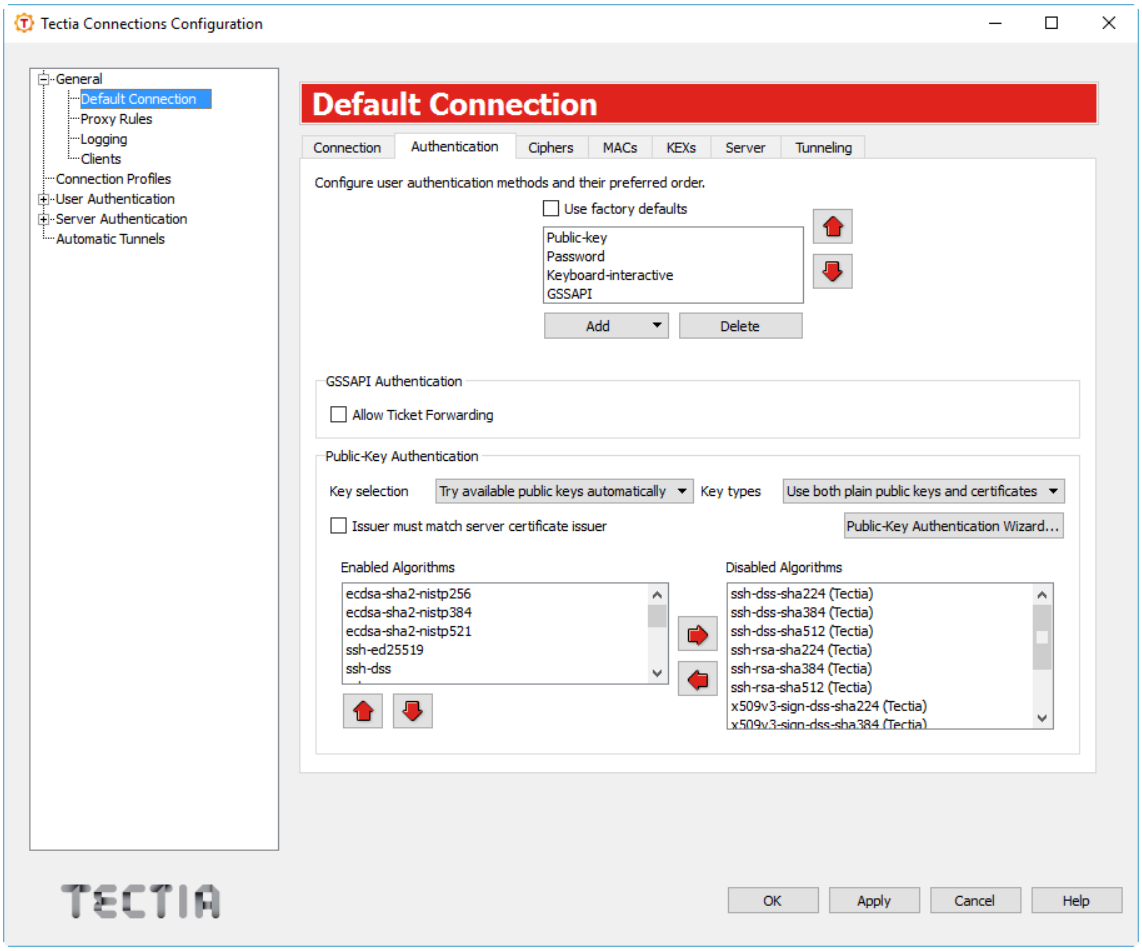


Figure A.4. Authentication methods for Tectia Client

Select the **Use factory defaults** check box to use the factory default authentication methods, or clear the check box to define a custom list of authentication methods.

In Tectia Client 7.0, the factory default authentication methods are, in order:

- Public-key

- Password
- Keyboard-interactive
- GSSAPI

The authentication methods are supported on all platforms, except for GSSAPI, which is not available on IBM z/OS.

To add a new authentication method to the list, click **Add** and select the method from the drop-down menu.

To remove an authentication method, select the method from the list and click **Delete**.

Use the arrow buttons to organize the preferred order of the authentication methods. The first method that is allowed by the Secure Shell server is used. Note that in some cases, the server may require several authentication methods to be passed before allowing login.

Possible methods for user authentication are:

- **Public-key**: Users are requested to use public-key authentication. See also Section A.1.4.
- **Password**: Users are requested to enter a password for authentication.
- **Keyboard-interactive**: Keyboard-interactive is designed to allow the Secure Shell client to support several different types of authentication methods, including RSA SecurID, and PAM. For more information on keyboard-interactive, see Section 4.8.
- **GSSAPI**: GSSAPI (Generic Security Service Application Programming Interface) is a common security service interface that allows different security mechanisms to be used via one interface. For more information on GSSAPI, see Section 4.9.

In the **GSSAPI Authentication** field, by selecting the **Allow Ticket Forwarding** check box you can enable Tectia Client to allow forwarding the Kerberos ticket over several connections.

When using **Public-Key Authentication**, you can also define which key types are used and how the keys are selected.

Key selection defines the policy Connection Broker uses when proposing user public keys to the server. Select the mode from the drop-down list. The options are:

- **Try available public keys automatically** (the default). With this policy, the client will try the keys in the following order:

  1. Keys with public key available and private key without a passphrase (no user interaction)
  2. Keys with public key available but private key behind a passphrase (require a passphrase query, provided the key is accepted by the server)
  3. The rest of the keys, meaning keys that require a passphrase for the public key as well as the private key.

- **Prompt user to select the public key** - with this policy, the Connection Broker prompts the user to select the key from a list of available keys. If authentication with the selected key fails, the client will prompt the user again to select another key.

Key types defines whether only plain public keys or only certificates are tried during public-key authentication. Select the key type from the drop-down list. The default is to try both plain public keys and certificates.

By selecting the **Issuer must match server certificate issuer** check box, you can make the Connection Broker filter the user certificates that will be included in the list presented to the user. The client-side user certificates can be filtered according to their issuer name that is compared to the certificate issuers requested or accepted by the server. By default, the filtering is not done. This option is useful when a user has several certificates with different access rights to the same server, for example for a testing role and for an administrator role. The Connection Broker chooses the relevant certificates that are applicable on the remote host, and the user can choose the correct certificate from the short-listed ones.

To generate new public-key pairs and to upload the public part of the key to a server, click the **Public-Key Authentication Wizard** button. For more information, see the section called “Using the Public-Key Authentication Wizard”.

Enabled algorithms lists the public-key signature algorithms that are used for authenticating and signing the user's public key. The algorithms that will be used are those that are configured for both Tectia Server and the Connection Broker. You can use the up and down arrow buttons to modify the order of the algorithms. To move an algorithm to the **Disabled algorithms** list, select it and click the right arrow button.

A list of factory default public-key signature algorithms can be found in Section E.4.

Defining Ciphers

On the **Ciphers** tab, you can define the encryption algorithms used.

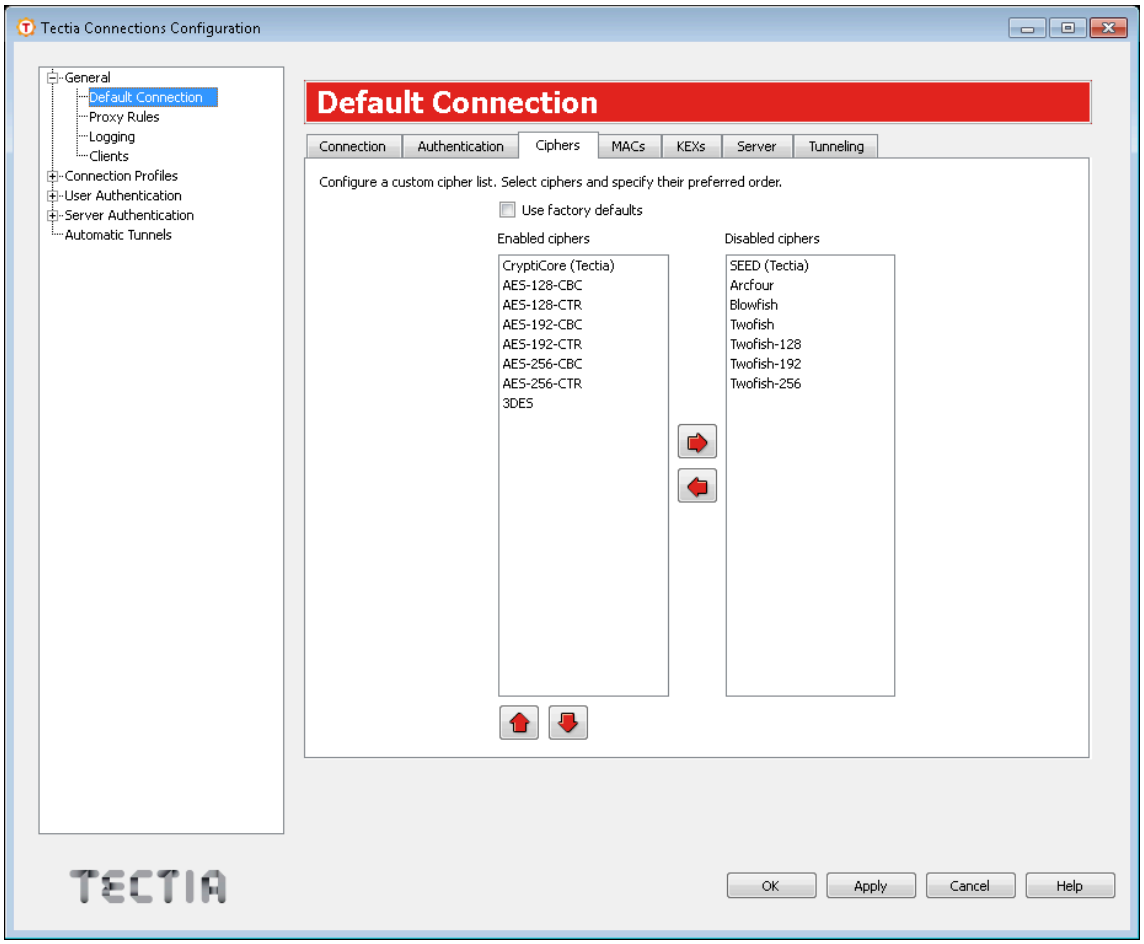


Figure A.5. Defining a cipher list

Select the **Use factory defaults** check box to use the factory default algorithms, or define a cipher list using the arrow buttons. The ciphers are tried in the order they are specified.

A list of factory default ciphers can be found in Section E.1.

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 Connection Broker configuration file.

Defining MACs

On the **MACs** tab, you can configure the message integrity algorithms used.

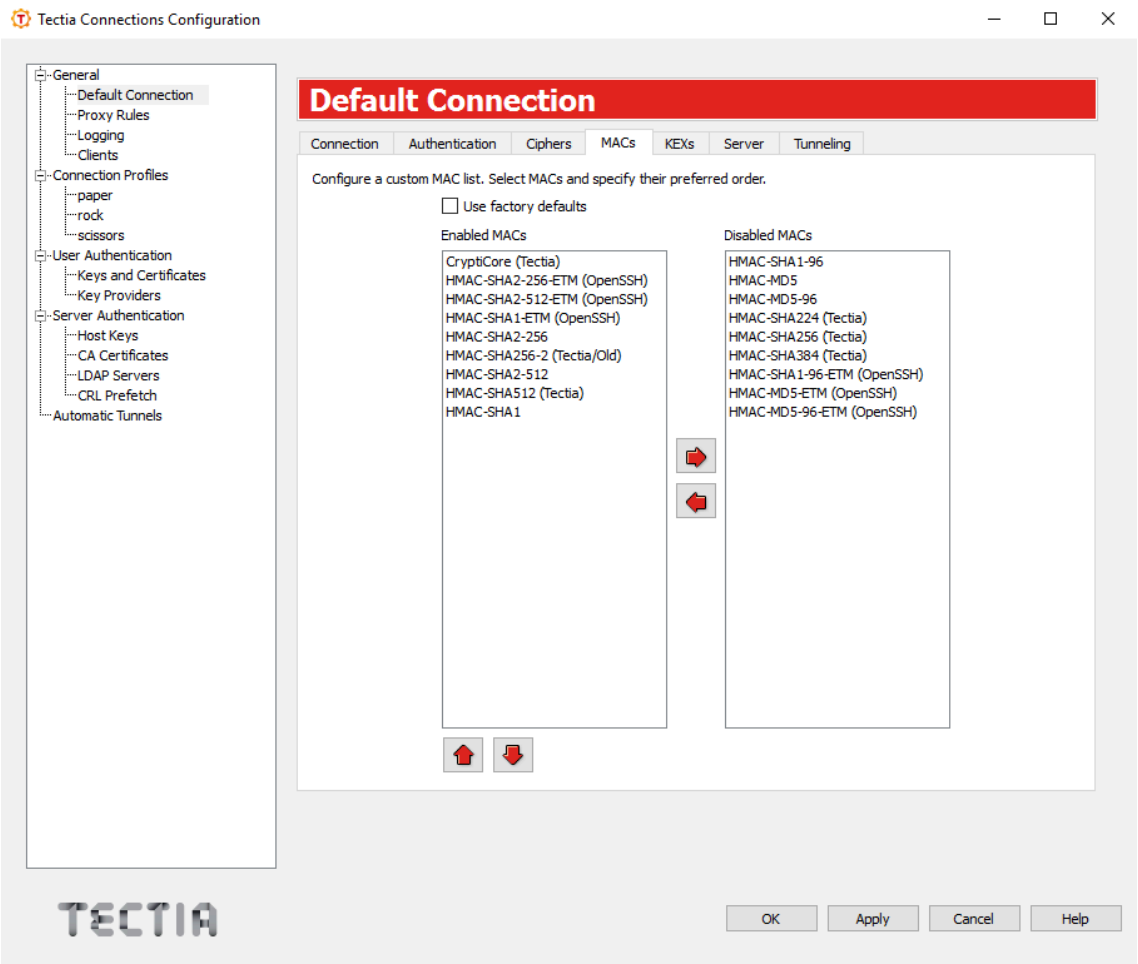


Figure A.6. Defining a MAC list

Select the **Use factory defaults** check box to use the factory default algorithms, or define a MAC list using the arrow buttons. The MACs are tried in the order they are specified.

A list of factory default MACs can be found in Section E.3

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 Connection Broker configuration file.

The algorithms marked with **(OpenSSH)** correspond to the algorithms that end with `@openssh.com` in the Connection Broker configuration file.

Defining KEXs

On the **KEXs** tab, you can configure the key exhange methods used.

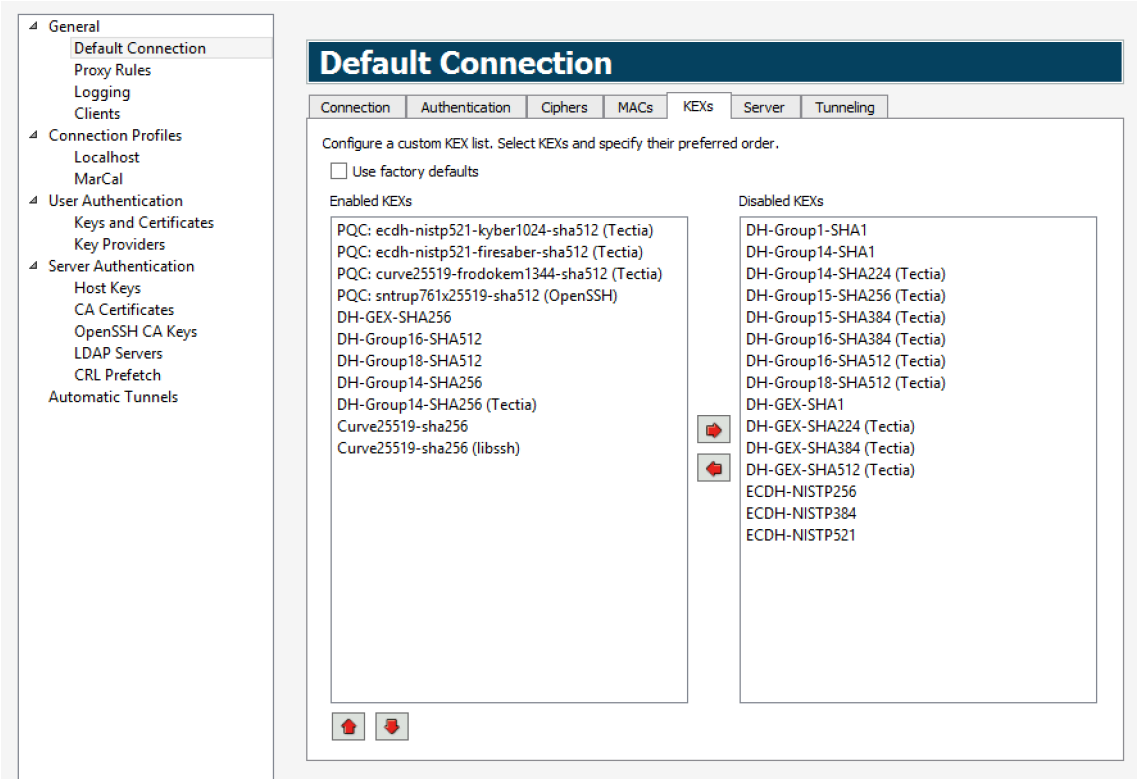


Figure A.7. Defining a KEX list

Select the **Use factory defaults** check box to use the factory default methods, or define a KEX list using the arrow buttons. The KEX methods are tried in the order they are specified.

A list of factory default KEXs can be found in Section E.2.

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 Connection Broker configuration file.

Defining Server Connections

On the **Server** tab, you can define advanced server connection settings.

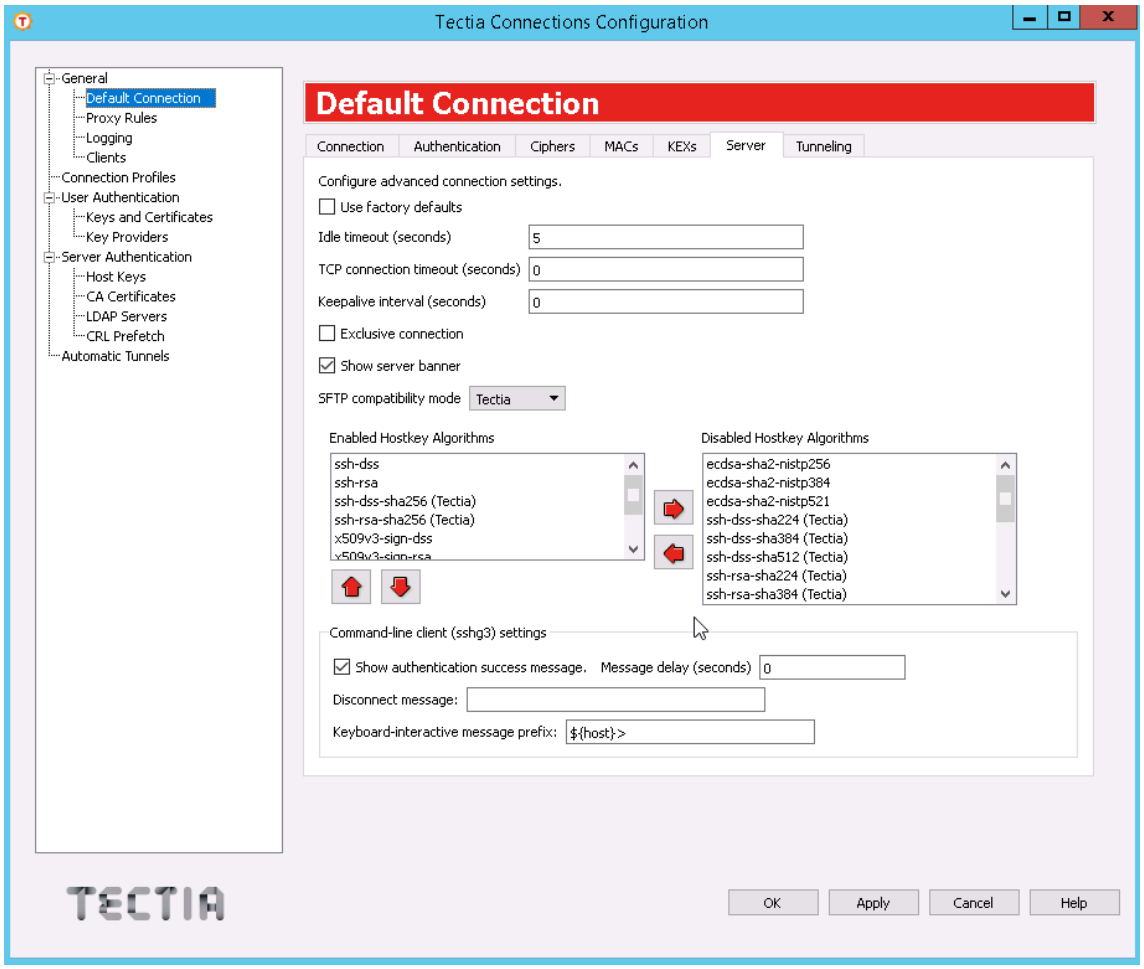


Figure A.8. Defining server connection settings

Use factory defaults

Select the check box to use the default values for the server connection settings.

Idle timeout

Specify how long idle time (after all connection channels are closed) is allowed for a connection before automatically closing the connection. The default is 5 seconds. Setting a longer time allows the connection to the server to remain open even after a session (for example, PrivX Desktop GUI) is closed. During this time, a new session to the server can be initiated without re-authentication. Setting the time to 0 (zero) terminates the connection immediately when the last channel to the server is closed.

TCP Connection Timeout

Specify for how long a TCP connection will be attempted to a Secure Shell server. Define the timeout in seconds, and after that time the TCP connection will be released in case the remote server is down or unreachable. Setting the value as 0 (zero) means this Tectia setting is disabled and the system default TCP timeout will be used. By default, the system timeout is used.

Keepalive interval

Specify an interval (in seconds) for sending keepalive messages to a Secure Shell server. The default is 0, meaning that no keepalive messages are sent.

Exclusive connection

Select this check box if you want always a new connection opened, instead of reusing a currently open connection.

Show server banner

Select this check box if you want to have the server banner message file (if it exists) visible to users before login.

Show authentication success message

Clear this check box if you do not want to have the `AuthenticationSuccessMsg` messages output and logged. By default the messages are enabled.

Message delay (seconds)

Sets the time for how long the authentication success message is shown. Default value is `2`. When the value is set to `0`, the message is not shown, only logged.

Disconnect message

Sets the message that is shown when you disconnect. By default this is turned off. For a list of accepted substitution variables, see **disconnect-message**

Keyboard-interactive message prefix

String shown before the keyboard-interactive prompt. Default value is `${host}>` . For a list of accepted substitution variables, see **keyboard-interactive**

SFTP compatibility mode

Select a suitable mode for transferring files with SFTP. This setting affects the behavior of the **get/mget/sget** and **put/mput/sput** commands and the recursion level used by the **sftpg3** client. The following options are available:

- `Tectia` (the default) - **sftpg3** transfers files recursively from the current directory and all its subdirectories.
- `OpenSSH` - copies only regular files and symbolic links from the specified directory, and no subdirectories are copied. Otherwise the semantics of the **get** command are unchanged.
- `FTP` - the **get/put** commands are executed as **sget/sput** meaning that they transfer a single file, and no subdirectories are copied.

The recursion depth can be overridden by using the **sftpg3** client's commands **get/put/mget/mput** with command-line option `--max-depth="LEVEL"`. For more information, see sftpg3(1).

Enabled Hostkey Algorithms

The host key signature algorithms used for server authentication with host keys or certificates are listed here. The algorithms that will be used are those that are defined in both Tectia Server and Connection Broker configuration files. This way the use of only certain algorithms, such as SHA-2, can be enforced by the server.

The host key algorithms are tried in the order they are specified. Exception: If a host key of a server already exists in the host key store of the client, its algorithm is preferred. You can use the up and down arrow buttons to modify the order of the algorithms.

A list of factory default host key algorithms can be found in Section E.4.

Disabled Hostkey Algorithms

The host key algorithms listed here are not used for server authentication. To disable a host key algorithm, select it in the **Enabled Hostkey Algorithms** list and click the right arrow button.

Defining Default Tunneling Settings

On the **Tunneling** tab, you can define the default settings for X11 connections and agent forwarding (tunneling). The defaults are applied to new connection profiles and to those connection profiles that do not have their own tunneling settings defined.

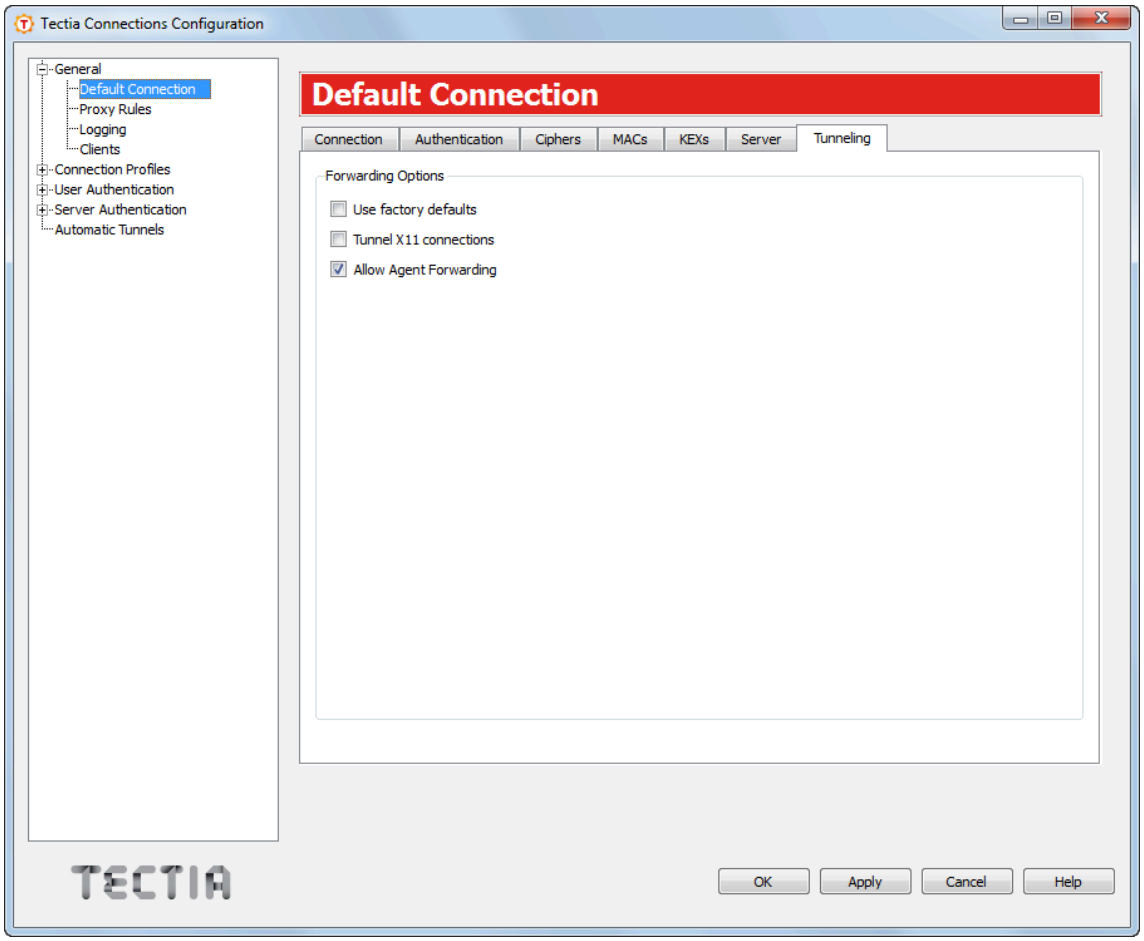


Figure A.9. Defining default tunneling settings

Select the **Use factory defaults** check box to apply the factory defaults for X11 and agent forwarding. According to the factory defaults, X11 forwarding is disabled (off) and agent forwarding is enabled (on).

To allow X11 forwarding on the client side, select the **Tunnel X11 connections** check box.

To disable agent forwarding on the client side, unselect the **Allow Agent Forwarding** check box.

Defining Proxy Rules

On the **Proxy Rules** page, you can define proxy rules to be used for connections.

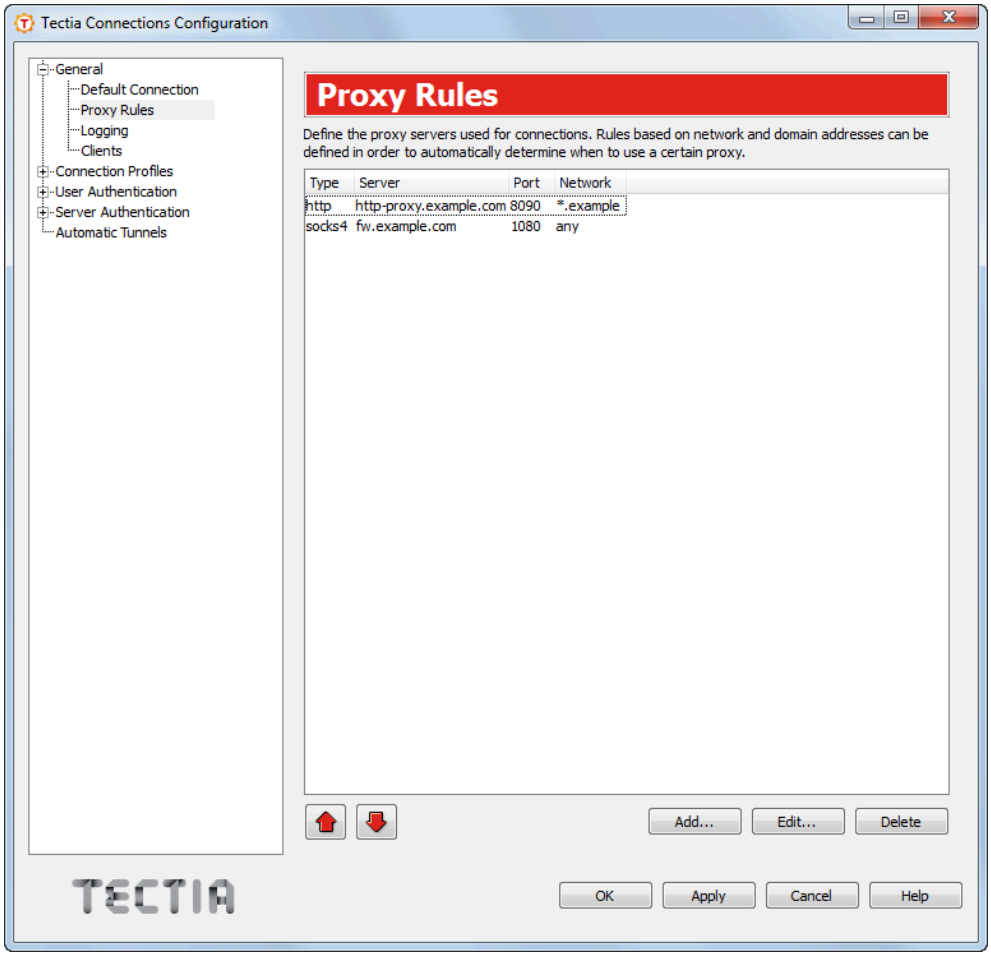


Figure A.10. Defining proxy rules

To add a new proxy rule:

1. Click **Add**. The **Proxy Rule** dialog box opens.

2. Select the **Type** of the rule. The type can be **Direct** (no proxy), **Socks4**, **Socks5**, or **Http**.

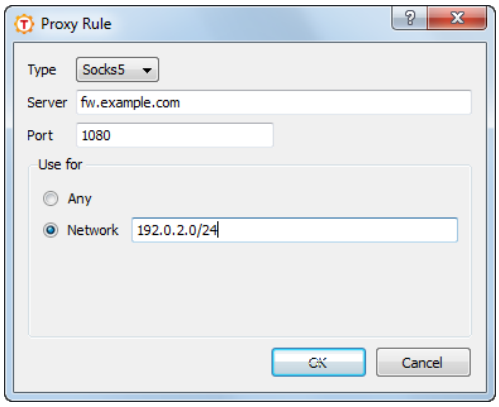


Figure A.11. Defining proxy settings

For other types than direct, enter the proxy **Server** address and **Port**.

Select also whether the proxy rules applies to **Any** connection or only to connections to the specified **Network**. In the **Network** field, you can enter one or more conditions delimited by commas (`,`). The conditions can specify IP addresses or DNS names.

The IP address/port conditions have an address pattern and an optional port range (`ip_pattern[:port_range]`).

The `ip_pattern` may have one of the following forms:

- a single IP address `x.x.x.x`
- an IP address range of the form `x.x.x.x-y.y.y.y`
- an IP sub-network mask of the form `x.x.x.x/y`

The DNS name conditions consist of a hostname which may be a regular expression containing the characters "*" and "?" and a port range (`name_pattern[:port_range]`).

Click **OK**.

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

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

The rules are read from top down. Use the arrow button to change the order of the rules.

To use these general proxy rules with a connection profile, you must select to do so in the profile settings. See the section called “Defining Proxy Settings”.

Defining Logging Settings

On the **Logging** page, you can enable logging and customize the information that will be logged in the event log. By default logging is disabled.

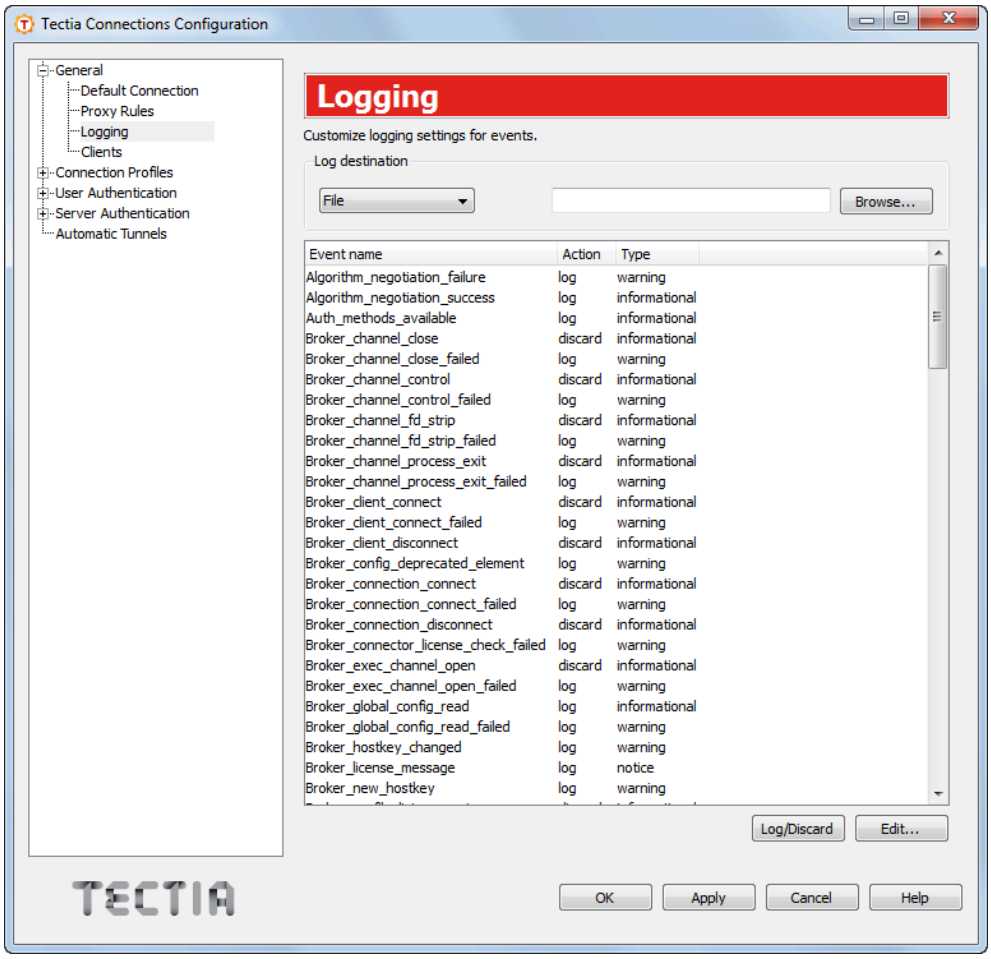


Figure A.12. Logging settings

To enable logging of Tectia Client internal events, select how the logs will be saved. In the **Log Destination** field:

- Select **File** to have the log data saved in to a file named in the field on the right. Enter the exact file name or browse to an existing file.
- Select **Event Log** to have the Tectia Client data stored in the Event Log of the host.

Each program-internal event has an associated **Action** and **Type**. They have reasonable default values, which are used if no explicit logging settings are made.

The action can be either **log** or **discard**.

The event type can be one of the following:

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

- **Security failure**

For a description of the log events, see Appendix 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 type, select an event from the list and click **Edit**. You can select multiple events by holding down the SHIFT or CTRL key while clicking. The **Edit Audit** dialog box opens. Select the **Action** (`log` or `discard`) and the **Type** (`informational`, `warning`, `error`, `security-success` or `security-failure`) for the event and click **OK**.

Defining Clients Settings

On the **Clients settings** page, you can define settings related to clients.

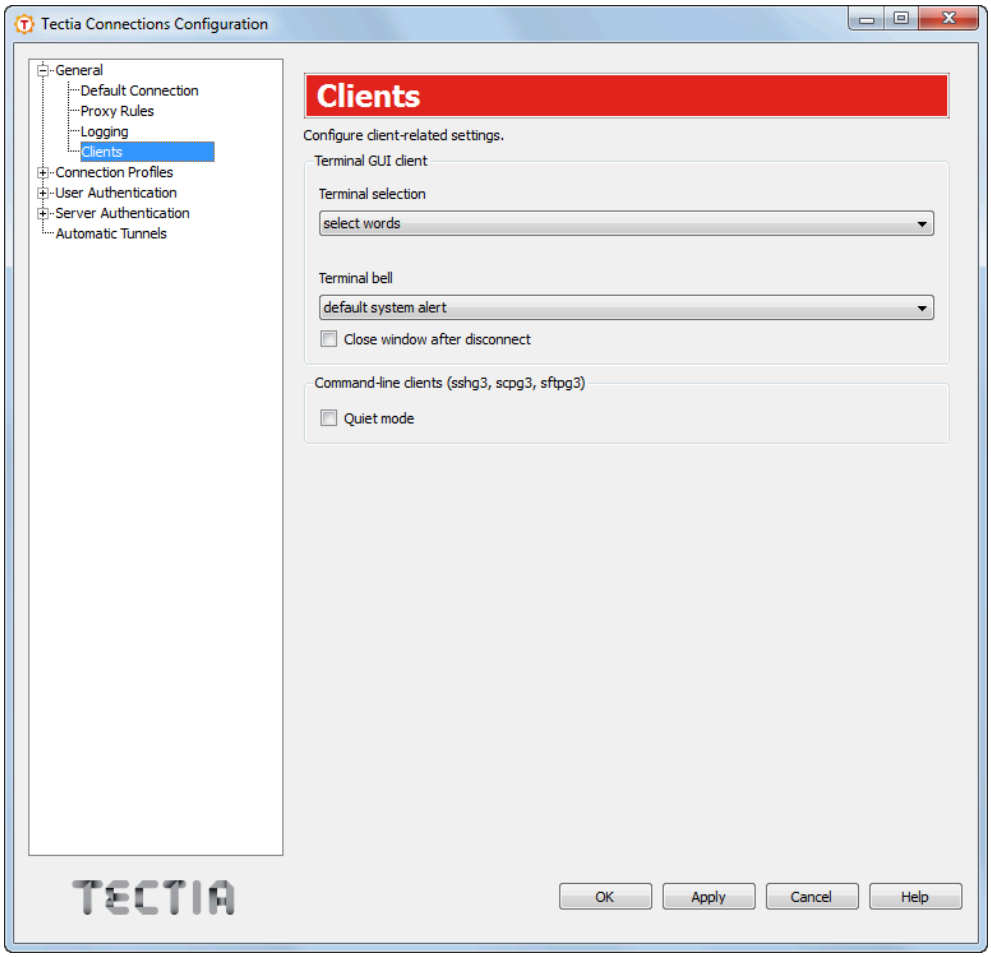


Figure A.13. Client settings

GUI client

Select the **Close window after disconnect** option to define that also the PrivX Desktop GUI window is to be closed while disconnecting from a server session by pressing `CTRL+D`. By default the terminal remains open, and only the server connection is closed.

Command line clients

The **Quiet mode** setting defines whether the command line clients should suppress warnings, error messages and authentication success messages. The setting affects the command line tools **scpg3**, **sshg3** and **sftpg3**.

A.1.3 Defining Connection Profiles

Under **Connection Profiles** you can configure separate connection settings for each Secure Shell server you connect to. You can also configure several profiles for the same server, for example, with different user accounts.

Click **Test Connection** to open a connection to the remote server. You need to connect to the server once in order to get the server's host key. Tectia Client will prompt you to verify the received key. Check that it is valid, preferably by calling the server's administrator, and save the validated key. After this, the locally saved information on the key will be used in the authentication process automatically.

- To add a connection profile, click **Add profile** in the **Connection Profiles** page. Enter a name for the profile and click **OK**. By default, the profile name is also used as the hostname of the server.

  Newly created connection profiles will inherit the default values for authentication, ciphers, MACs, KEXs, and advanced server settings defined under the **General → Defaults** page (the section called "Defining Default Connection Settings"). The values can be customized on the profile-specific tabbed pages.

  Define the profile settings in the tabbed view as described in the section called "Defining Connection Settings", the section called "Defining Authentication", the section called "Defining Ciphers", the section called "Defining MACs", the section called "Defining KEXs", the section called "Defining Server Connections", the section called "Defining Proxy Settings", the section called "Defining Tunneling",

- You can organize the connection profiles in folders for each server you are connecting to. To add a folder for connection profiles, click **Add folder** in the **Connection Profiles** page. Enter a name for the folder and click **OK**. Add connection profiles to the folder by selecting the folder and clicking **Add profile**. The profile is created into the folder.

- To move a profile to a different profile folder, select the profile from the list and click **Move**. Select the folder where you want to move the profile from the drop-down list and click **OK**.

- To rename a connection profile or a profile folder, right-click on a profile or a folder name under **Connection Profiles** and click **Rename**. Type a new name, press **Enter**, and click **OK** or **Apply**.

- To remove a connection profile or a profile folder, select a profile or a folder and click **Delete**. You will be asked for confirmation. Click **OK** to proceed with the deletion.

  Note that removing a profile folder removes also all profiles in it.

-

•

Defining Connection Settings

On the **Connection** tab, you can define the protocol settings used in the connection. Any changed connection settings will take effect the next time you log in.

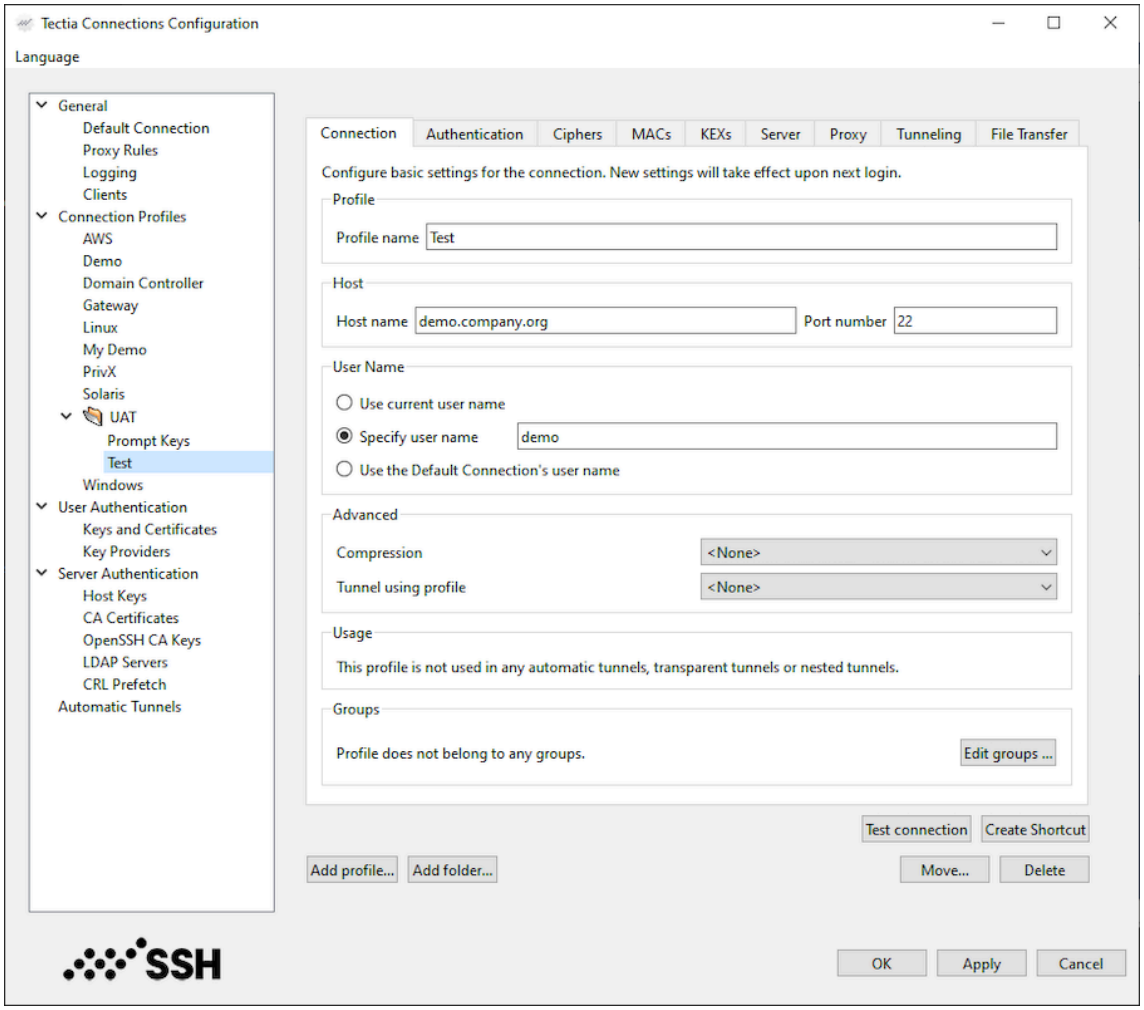


Figure A.14. Configuring connection profiles

Host Name

Specify the host name or the IP address of the remote host computer to which you want to connect with the profile.

Port Number

Define the listen port on the Secure Shell server. The default SSH port number is 22. In case you know that the remote server uses another port, enter the number in the **Port Number** field.

Note

A Secure Shell server program must be listening to the specified port on the remote host computer or the connection attempt will not succeed. If you are unsure which port the remote host computer is listening to, contact the system administrator of the remote host.

User Name

Select **Use current Windows user name** if the connection should always be made using the currently logged in Windows user name. This is similar to defining `%USERNAME%` (note the percent signs) as the user name. `%USERNAME%` reads the actual user name from an environment variable.

Select **Specify user name** and enter the user name, if you want to define the user name this profile will use when connecting to the remote host computer.

Select **Prompt user for the user name** if the user should enter the user name manually every time when connecting.

Select **Use the Default Connection's user name** if you want to apply the generic user name defined in the **General - Default Connection** settings.

Advanced

In **Compression**, select the desired compression setting from the drop-down menu. Valid choices are **zlib** and **none**. Compression is disabled by default.

In **Tunnel using profile**, use the drop-down list to select a profile for creating a nested tunnel. The first tunnel will be created to the server defined in the current connection profile, and from there, the second tunnel will be created to a host defined in the profile selected with the **Tunnel using profile** setting. By default, tunneling is disabled.

Usage

This field shows information on where the defined profile is used.

Defining Authentication

On the **Authentication** tab, you can define the user authentication methods for the profile.

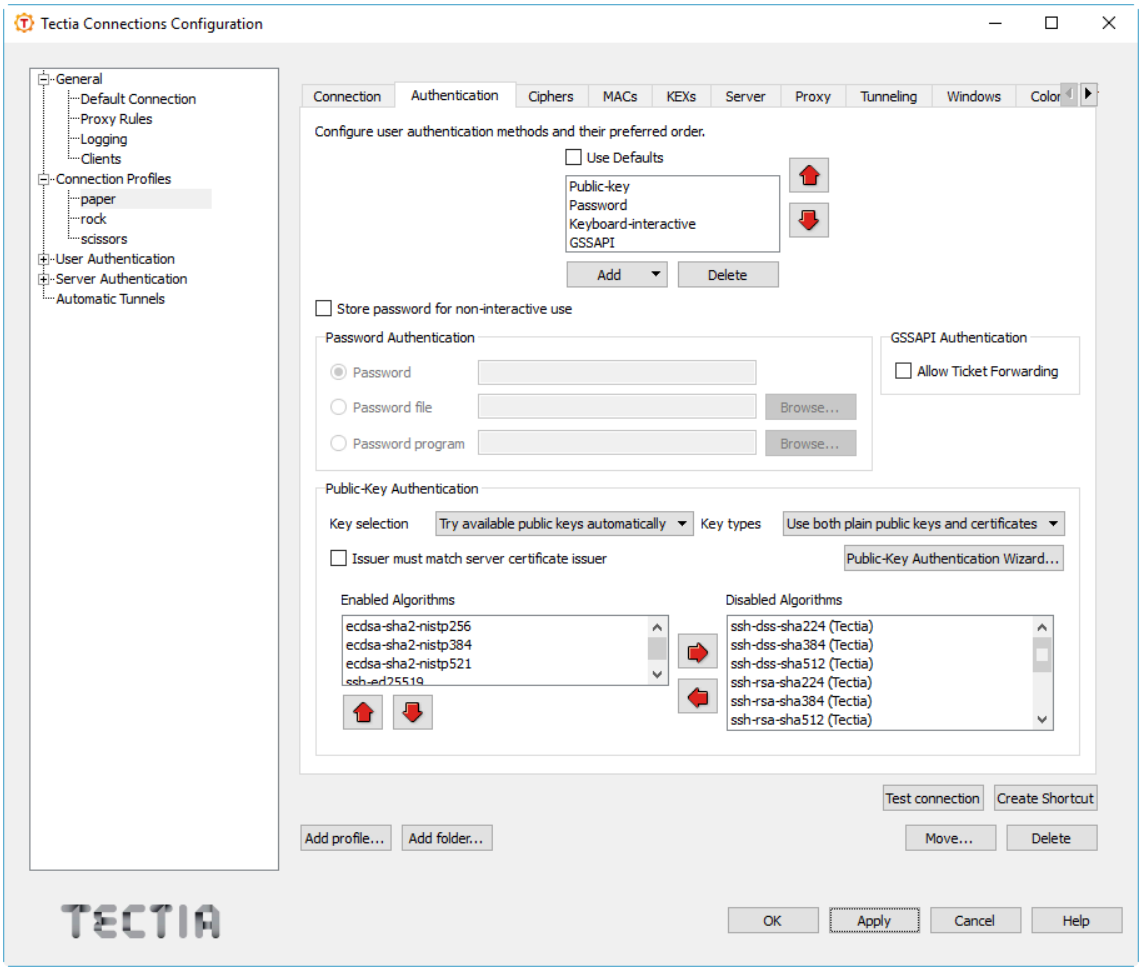


Figure A.15. Configuring authentication methods for the profile

1. Select the **Use Defaults** check box to use the authentication methods defined on the **Default Connection** page (the section called “Defining Authentication”), or clear the check box to define a custom list of authentication methods.

   To add a new authentication method to the list, click **Add** and select the method from the drop-down menu.

   To remove an authentication method, select a method from the list and click **Delete**.

   Use the arrow buttons to organize the preferred order of the authentication methods. The first method that is allowed by the Secure Shell server is used. Note that in some cases, the server may require several authentication methods to be passed before allowing login.

   Possible methods for user authentication are:

   - **Public-key**: Use public-key authentication. See also Section A.1.4.
   - **Password**: Use a password for authentication.

- **Keyboard-interactive**: Keyboard-interactive is designed to allow the Secure Shell client to support several different types of authentication methods, including RSA SecurID, and PAM. For more information on keyboard-interactive, see Section 4.8.

- **GSSAPI**: GSSAPI (Generic Security Service Application Programming Interface) is a common security service interface that allows different security mechanisms to be used via one interface. For more information on GSSAPI, see Section 4.9.

2. If you want to use the profile in non-interactive connections, you can select to store a password with the profile in the **Password Authentication** field.

   Select **Password** to enter the actual password string.

   Select **Password file** to enter a path to a file containing the password.

   Select **Password program** to enter a path to a program or a script that outputs the password.

   ## Caution

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

   ## Note

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

3. In the **GSSAPI Authentication** field, by selecting the **Allow Ticket Forwarding** check box you can enable Tectia Client to allow forwarding the Kerberos ticket over several connections.

4. When using **Public-Key Authentication**, you can also define which key types are used and how the keys are selected.

   Key selection defines the policy Connection Broker uses when proposing user public keys to the server. Select the mode from the drop-down list. The options are:

   - **Try available public keys automatically** (the default). With this policy, the client will try the keys in the following order:

     a. Keys with public key available and private key without a passphrase (no user interaction)

     b. Keys with public key available but private key behind a passphrase (require a passphrase query, provided the key is accepted by the server)

     c. The rest of the keys, meaning keys that require a passphrase for the public key as well as the private key.

- **Prompt user to select the public key** - with this policy, the Connection Broker prompts the user to select the key from a list of available keys. If authentication with the selected key fails, the client will prompt the user again to select another key.

Key types defines whether only plain public keys or only certificates are tried during public-key authentication. Select the key type from the drop-down list. The default is to try both plain public keys and certificates.

By selecting **Issuer must match server certificate issuer**, you can make the Connection Broker filter the user certificates that will be included in the list presented to the user. The client-side user certificates can be filtered according to their issuer name that is compared to the certificate issuers requested or accepted by the server. By default, the filtering is not done. This option is useful when a user has several certificates with different access rights to the same server, for example for a testing role and for an administrator role. The Connection Broker chooses the relevant certificates that are applicable on the remote host, and the user can choose the correct certificate from the short-listed ones.

To generate a public-key pair and to upload it to the remote server, click the **Public-Key Authentication Wizard** button. For instructions, see the section called “Using the Public-Key Authentication Wizard”.

Enabled algorithms lists the public-key signature algorithms that are used for authenticating and signing the user's public key. The algorithms that will be used are those that are configured for both Tectia Server and the Connection Broker. You can use the up and down arrow buttons to modify the order of the algorithms. To move an algorithm to the **Disabled algorithms** list, select it and click the right arrow button.

5. Click OK to save the connection profile.

Using the Public-Key Authentication Wizard

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

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

Select the **Keys and Certificates** page under **User authentication** and click **New Key** to start the Public-Key Authentication Wizard.

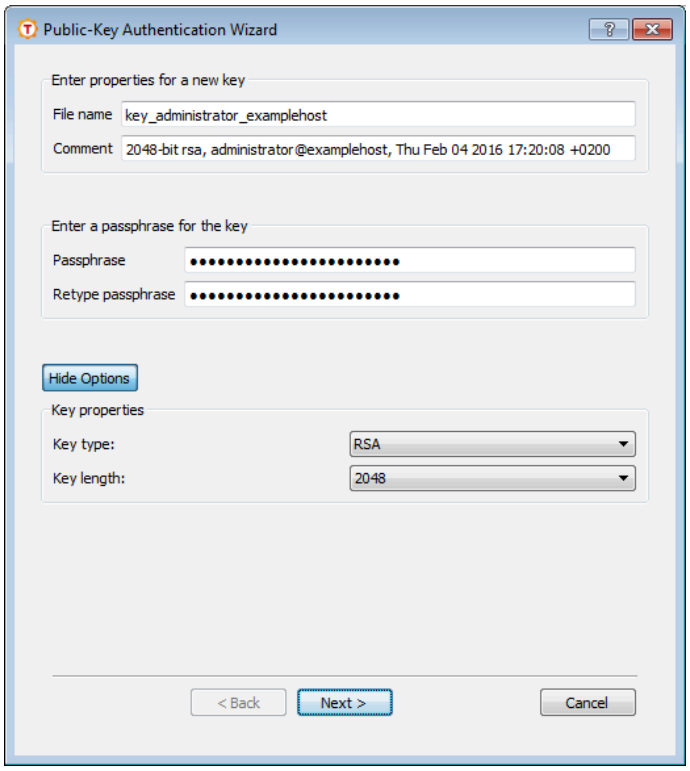


Figure A.16. The Public-Key Authentication Wizard

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

File Name

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

Comment

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

Passphrase

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

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.

Make the passphrase difficult to guess. Use ideally at least 20 characters, both letters and numbers. Any punctuation characters can be used as well. While the passphrase or private key are never sent

over the network, a dictionary attack can be used against a private key if it is accessible locally. For ease of use, an authentication agent is recommended instead of leaving the passphrase empty. By default ssh-broker-g3 functions as an authentication agent.

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

For connections where no user interaction is available, you can consider leaving the passprase empty.

Retype passphrase

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

Click the **Advanced Options**, to define the type of the key to be generated and the key length to be different from the defaults. By default, Tectia Client generates a pair of 3072-bit RSA keys.

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

Key Type

Select the type of the key to be generated. Available options are Ed25519, RSA, ECDSA and DSA.

Note

In FIPS mode (conforming to FIPS 186-5) RSA, ECDSA and Ed25519 are supported. DSA has been deprecated.

Key Length

Select the length (complexity) of the key to be generated. Available options are:

- DSA/RSA keys: 2048, 3072, 4096, 5120, 6144, 7168, 8192 bits
- ECDSA keys: 256, 384, 521 bits
- Ed25519 keys: 256 bits

Larger keys of the same key type are more secure, but also slower to generate. A 256-bit ECDSA key and a 3072-bit RSA key provide equivalent security.

As soon as a new key has been generated, the Wizard proceeds to uploading the key to a remote server. In case you want to upload an existing key to a remote server, select the key file in the Keys and Certificates view, and click **Upload**. The following dialog appears in both cases:

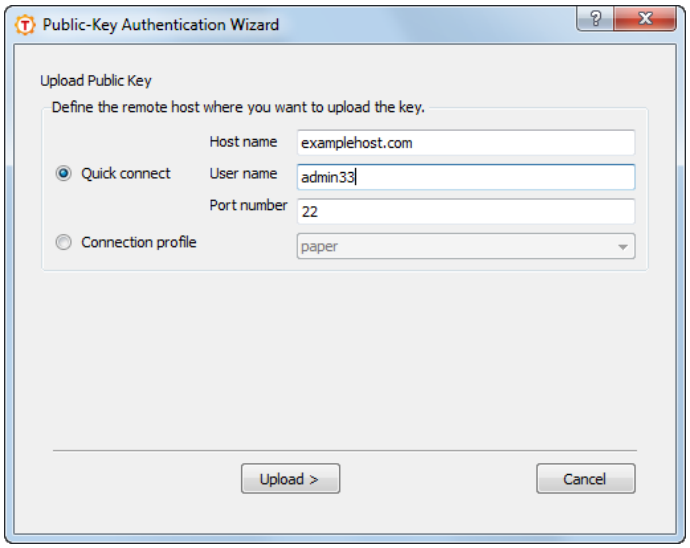


Figure A.17. Uploading a key

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

Quick connect

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

Connection profile

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

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

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

Defining Ciphers

On the **Ciphers** tab, you can define the encryption algorithms used for the profile.

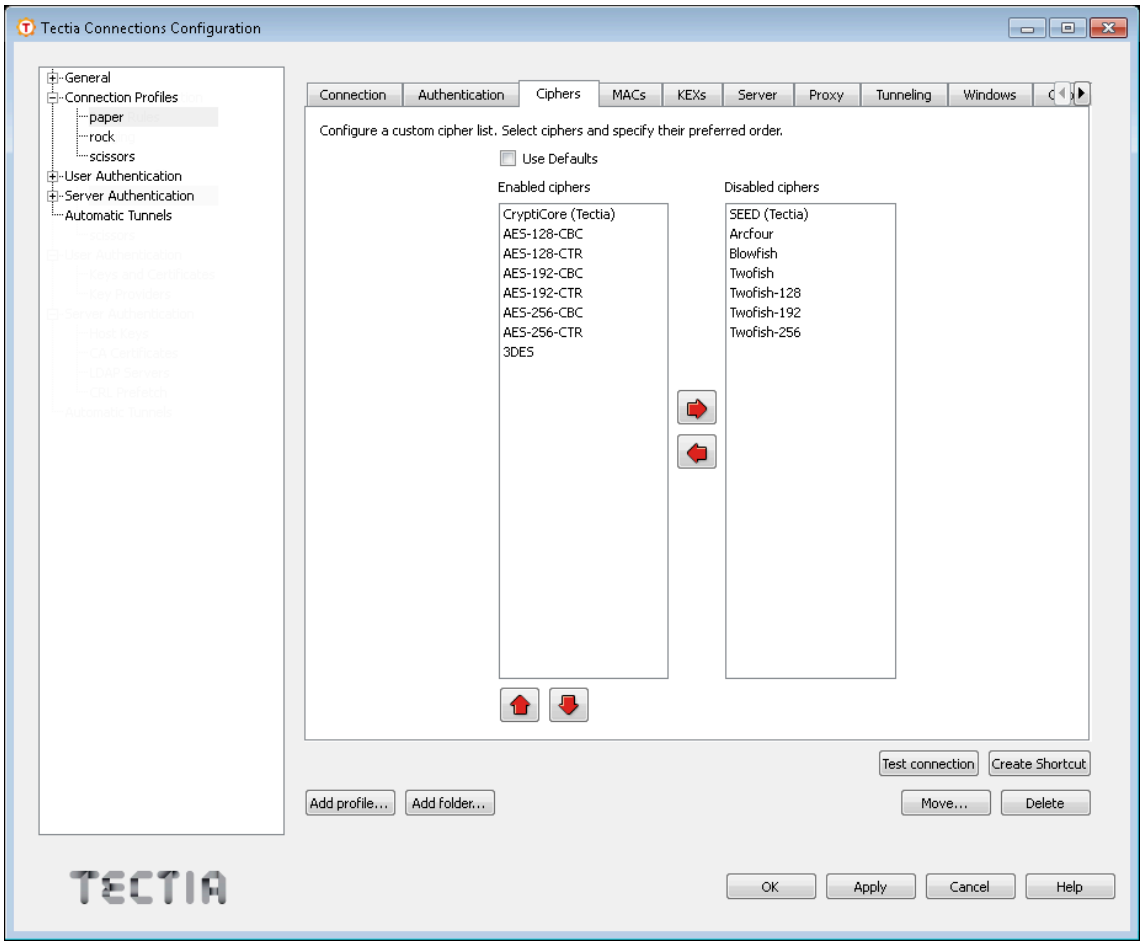


Figure A.18. Defining a cipher list for the profile

Select the **Use Defaults** check box to use the algorithms defined on the **Default Connection** page (the section called “Defining Ciphers”), or define a cipher list using the arrow buttons. The ciphers are tried in the order they are specified.

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 Connection Broker configuration file.

Defining MACs

On the **MACs** tab, you can configure the message integrity algorithms used for the profile.

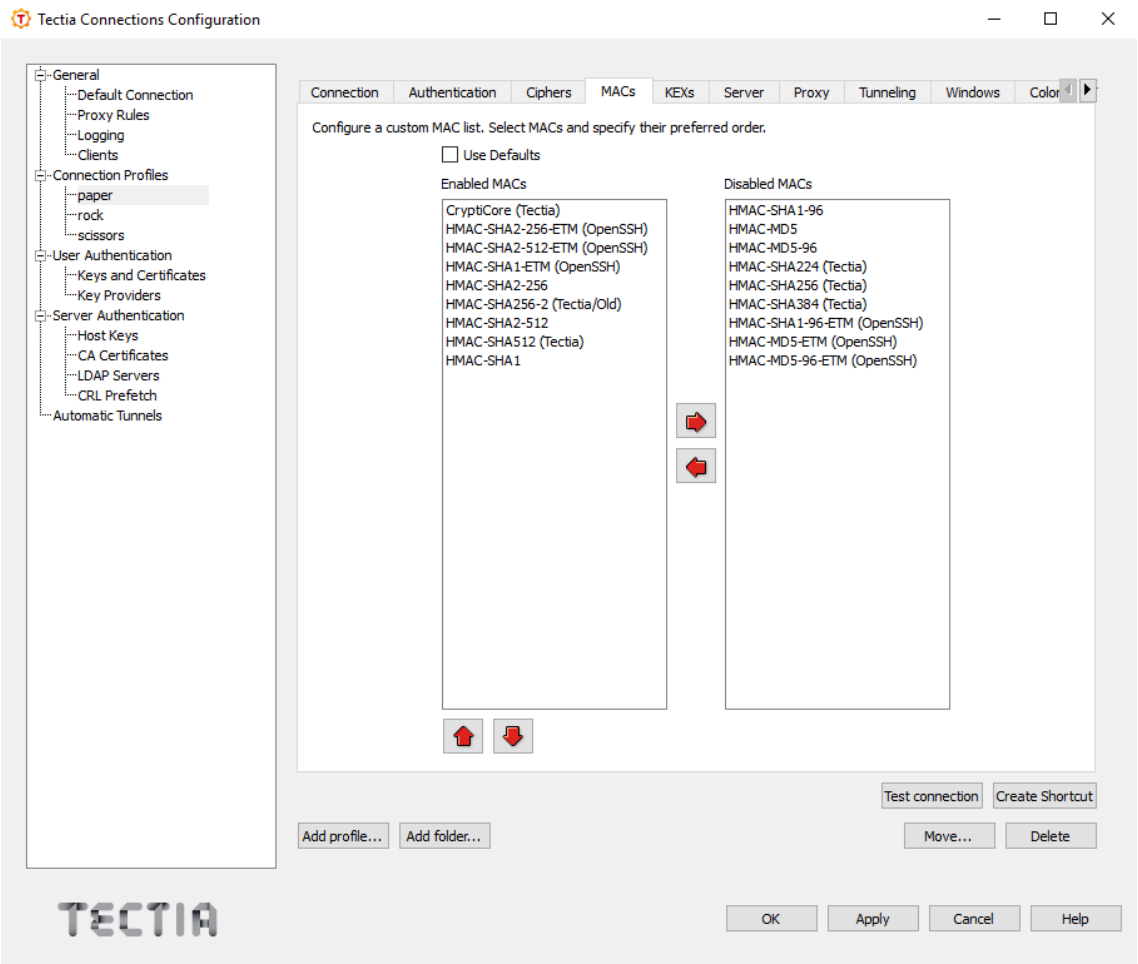


Figure A.19. Defining a MAC list for the profile

Select the **Use Defaults** check box to use the algorithms defined on the **Default Connection** page (the section called “Defining MACs”), or define a MAC list using the arrow buttons. The MACs are tried in the order they are specified.

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 Connection Broker configuration file.

Defining KEXs

On the **KEXs** tab, you can configure the key exchange methods used for the profile.

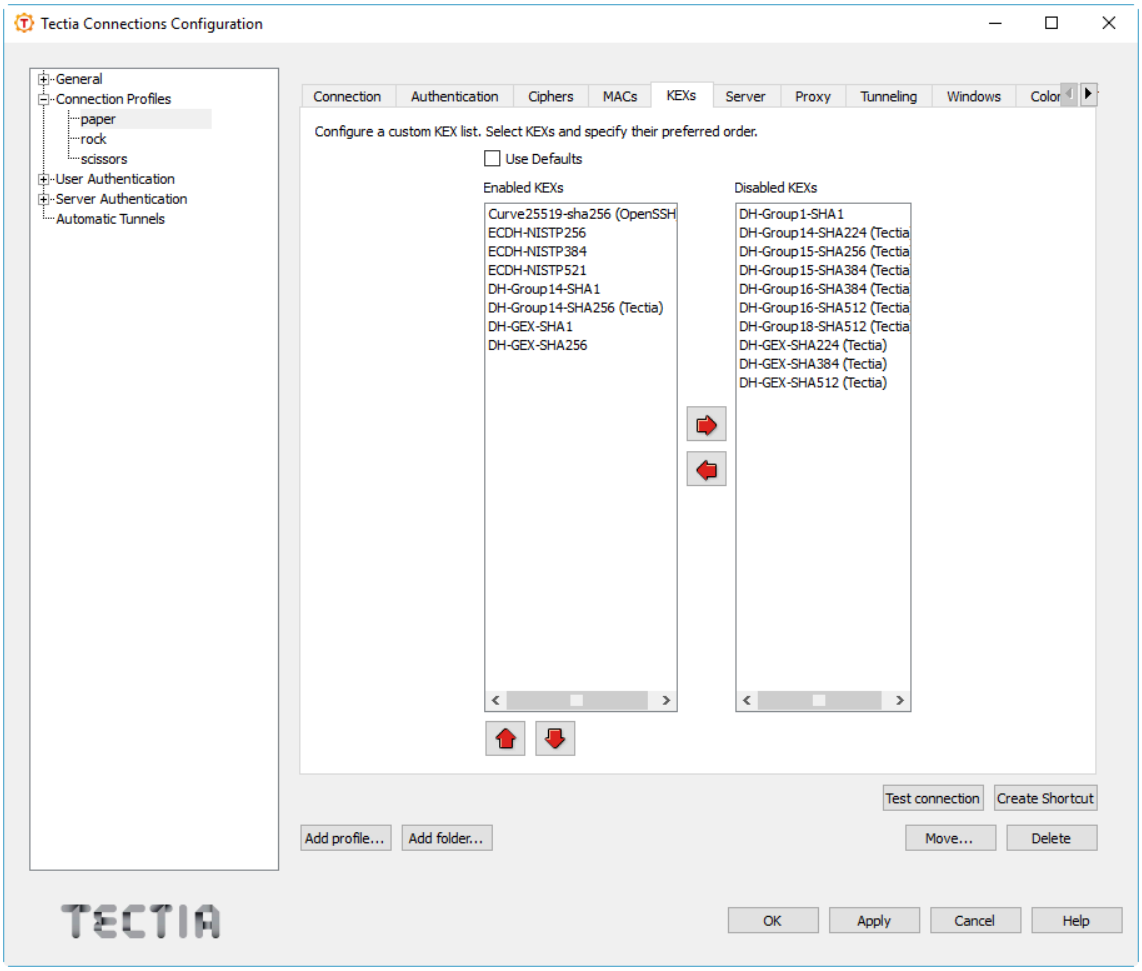


Figure A.20. Defining a KEX list for the profile

Select the **Use Defaults** check box to use the methods defined on the **Default Connection** page (the section called “Defining KEXs”), or define a KEX list using the arrow buttons. The KEXs are tried in the order they are specified.

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 Connection Broker configuration file.

Defining Server Connections

On the **Server** tab, you can define advanced server connection settings for the profile.

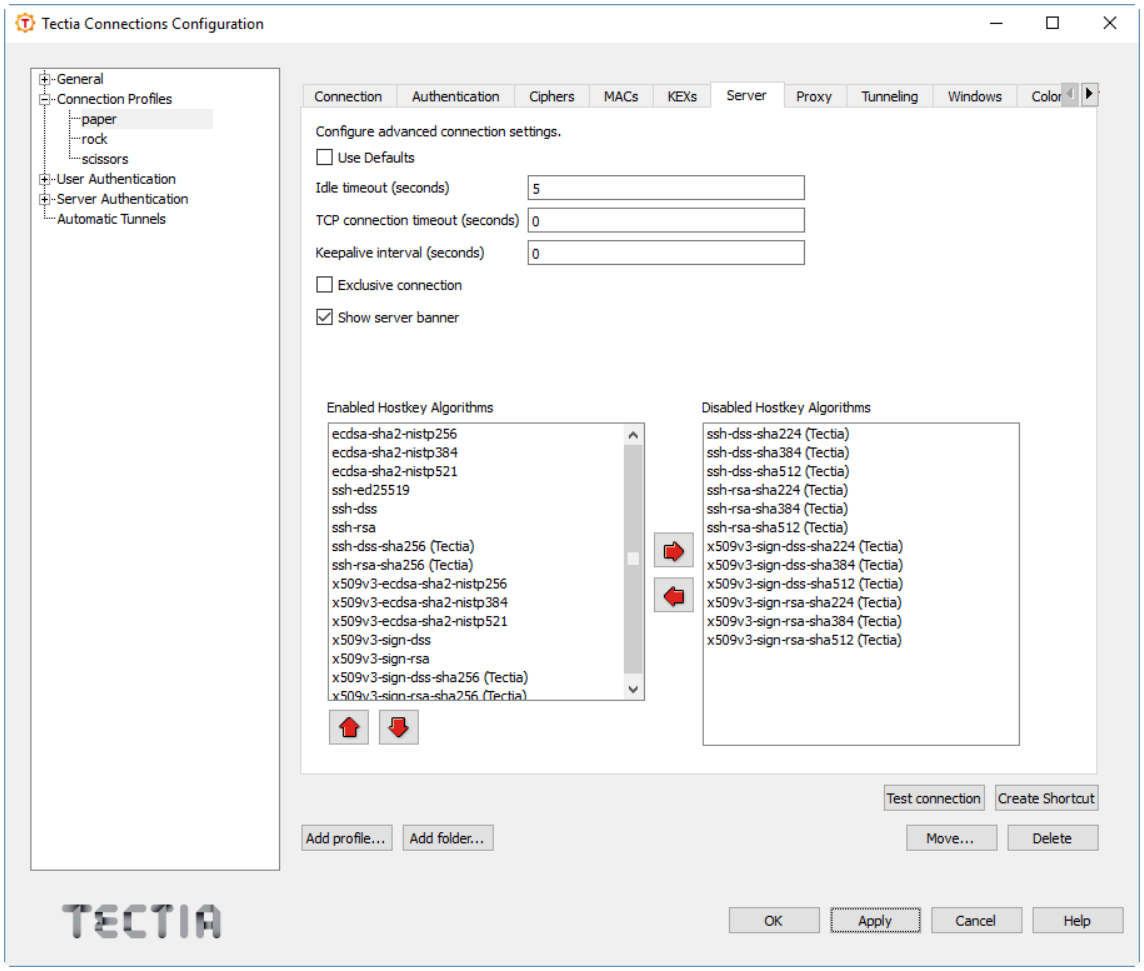


Figure A.21. Defining server connection settings for the profile

Use Defaults

Select the check box to use the values defined on the **Default Connection** page (the section called "Defining Server Connections") for the server connection settings.

Idle timeout

Specify how long idle time (after all connection channels are closed) is allowed for a connection before automatically closing the connection. The default is 5 seconds. Setting a longer time allows the connection to the server to remain open even after a session (for example, PrivX Desktop GUI) is closed. During this time, a new session to the server can be initiated without re-authentication. Setting the time to 0 (zero) terminates the connection immediately when the last channel to the server is closed.

TCP connection timeout

Specify for how long a TCP connection will be attempted to a Secure Shell server. Define the timeout in seconds. After the defined time the TCP connection will be released in case the remote server is

down or unreachable. Setting the value as 0 (zero) means that the default system TCP timeout will be used.

Keepalive interval

Specify an interval (in seconds) for sending keepalive messages to a Secure Shell server. The default is 0, meaning that no keepalive messages are sent.

Exclusive connection

Select this check box if you want that the profile always opens a new connection, instead of reusing a currently open connection.

Show server banner

Select the check box if you want to have the server banner message file (if it exists) visible to users before login.

Enabled Hostkey Algorithms

This list shows the host key signature algorithms used for server authentication with host keys or certificates. The algorithms that will be used are those that are defined in both Tectia Server and Connection Broker configuration files. This way the use of only certain algorithms, such as SHA-2, can be enforced by the server.

The host key algorithms are tried in the order they are specified, with one exception: If a host key of a server already exists in the host key store of the client, its algorithm is preferred. You can use the up and down arrow buttons to modify the order of the algorithms.

Disabled Hostkey Algorithms

The host key algorithms listed here are not used for server authentication. To disable a host key algorithm, select it in the **Enabled Hostkey Algorithms** list and click the right arrow button.

Defining Proxy Settings

On the **Proxy** tab, you can select proxy settings for the profile.

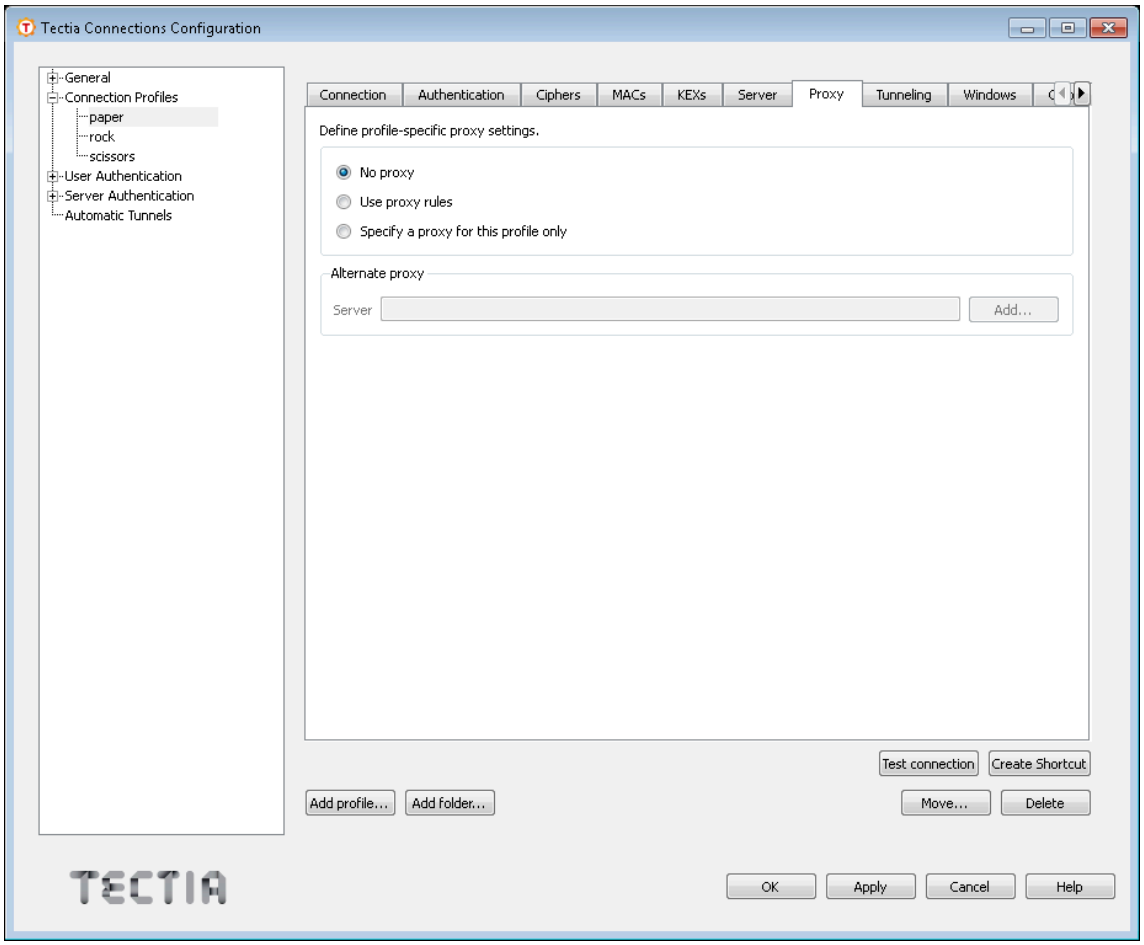


Figure A.22. Defining proxy settings for the profile

No proxy

Select this option if you do not want to use a proxy.

Use proxy rules

Select this option to use the proxy rules defined in the **General** settings **Proxy Rules** page (the section called “Defining Proxy Rules”).

Specify a proxy for this profile only

Click **Add** to add a new proxy definition for this profile.

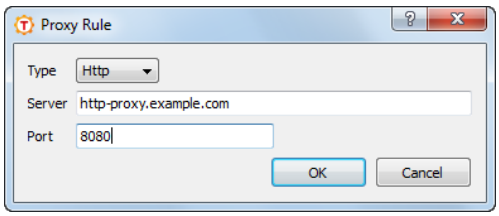


Figure A.23. Defining alternate proxy for the profile

Select the **Type** of the rule. The type can be **Direct**, **Socks4**, **Socks5**, or **Http**.

For other types than direct, enter the address of the proxy **Server** and **Port**.

Defining Tunneling

Tunneling, or port forwarding, is a way of forwarding otherwise unsecured TCP traffic through an encrypted Secure Shell connection (tunnel). You can secure for example POP3, SMTP, and HTTP connections that would otherwise be unsecured.

The tunneling settings for the connection profile are configured using the **Tunneling** tab. Any changed tunneling settings will take effect the next time you log in.

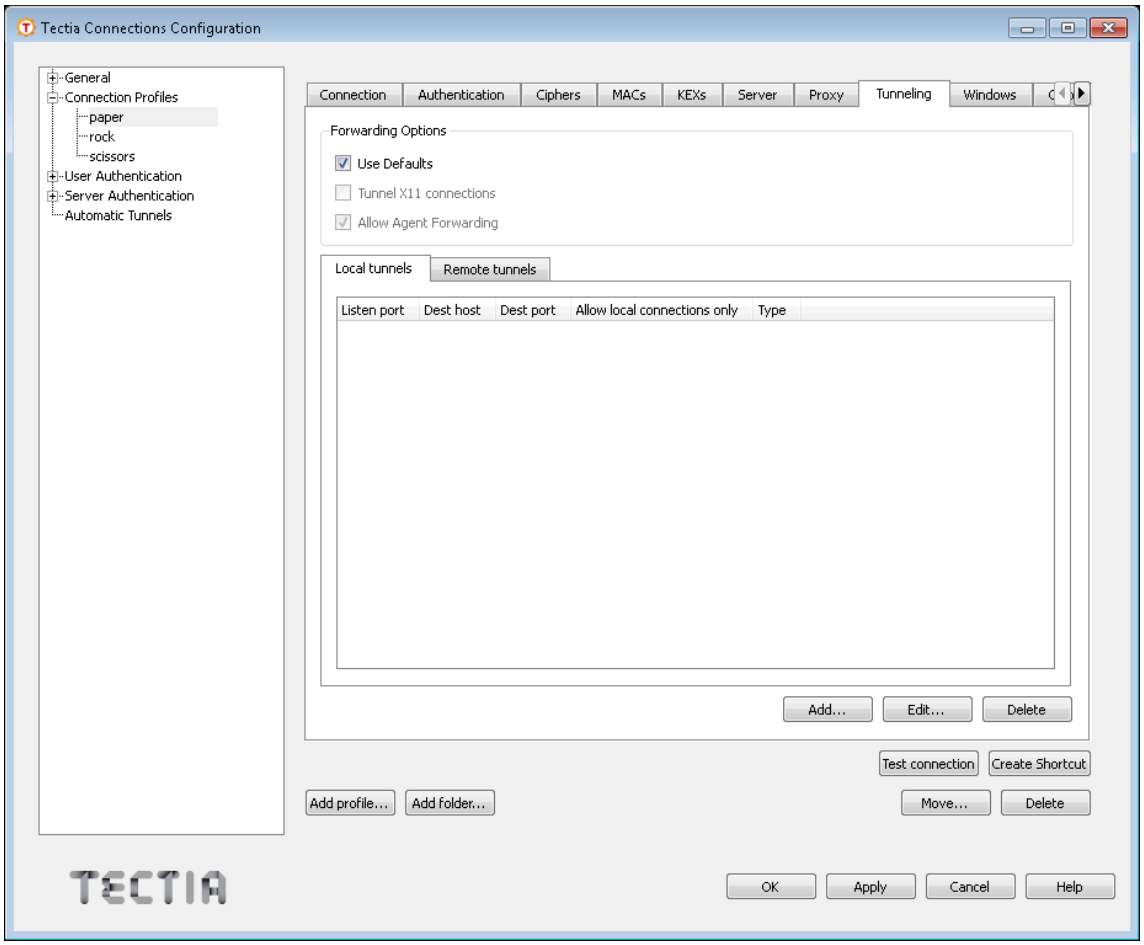


Figure A.24. Defining tunneling through a profile

Note

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

Forwarding Options

It is possible to define separately for each connection profile whether X11 and/or agent forwarding are enabled, or whether the general default forwarding settings are applied to the profile.

Use Defaults

Select this option to make the profile follow the default settings for X11 and agent forwarding defined on the **Defaults - Tunneling** tab (the section called “Defining Default Tunneling Settings”).

Tunnel X11 connections

To allow X11 forwarding for this connection profile, select this check box.

Tectia Client can securely tunnel (forward) X11 graphic connections from the remote host computer to an X Windows server running on the local computer.

Note

A prerequisite for X11 tunneling is that you have an X emulator (such as eXceed or Reflection X) running in passive mode on the Windows computer.

To tunnel (forward) X11 traffic, do the following actions:

1. Install an X server (X emulation) program on Windows (eXceed, Reflection X, or the like).
2. Start Tectia Client.
3. Select the **Tunneling** tab of the Connection Profiles page and make sure that the **Tunnel X11 connections** check box is selected.
4. Save your settings for Tectia Client.
5. Restart Tectia Client and log into the remote host.
6. Start the X server (X emulation) program.
7. To test the tunneling, run xterm or xclock from Tectia Client.

For more information, see Section 6.3.

Allow Agent Forwarding

To allow agent forwarding on the client side for this connection profile, select this check box.

In agent forwarding, Secure Shell connections and public-key authentication data are forwarded from one server to another without the user having to authenticate separately for each server.

For more information, see Section 6.4.

Local Tunnels

There are two types of tunnels that can be defined for application tunneling, local (outgoing) tunnels and remote (incoming) tunnels.

Local tunnels protect TCP connections that your local computer forwards from a specified local port to a specified port on the remote host computer you are connected to. It is also possible to forward the connection beyond the remote host computer, but the connection is encrypted only between Tectia Client and Tectia Server.

Remote tunnels protect TCP connections that a remote host forwards from a specified remote port to a specified port on your local computer.

To edit local tunnel definitions, click the **Local tunnels** tab.

To add a new local tunnel, click **Add**. The **Local Tunnel** dialog box opens.

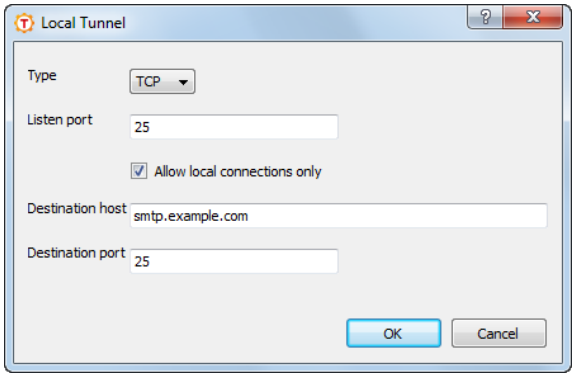


Figure A.25. Defining a local tunnel

The following fields are used to define a local tunnel:

- **Type**: Select the type of the tunnel from the drop-down list. Valid choices are TCP and FTP. If you are tunneling an FTP connection, set the tunnel type as FTP. For other protocols, set the tunnel type as TCP.

  Note

  If the Secure Shell server and the FTP server are located on different computers, FTP tunneling works only if FTP is set to run in passive mode. If the Secure Shell server and the FTP server are located on the same computer, tunneling works regardless of whether FTP is running in passive or active mode. For more information on FTP tunneling, see Section 6.1.3.

- **Listen port**: This is the number of the local port which the tunnel listens to or captures.

  Note

  The protocol or application that you wish to create the tunnel for may have a fixed port number (for example 143 for IMAP) that it needs to use to connect successfully. Other protocols or

applications may require an offset (for example 5900 for VNC) that you will have to take into an account.

- **Allow local connections only**: Select this option if you want to allow only local connections to be made. This means that other computers will not be able to use the tunnel created by you. By default, only local connections are allowed. This is the right choice for most situations.

  Consider the security implications carefully if you decide to also allow outside connections.

- **Destination host**: This field defines the destination host for the tunneling. The default value is `localhost`.

Note

The destination host is resolved by the Secure Shell server, so here `localhost` refers to the Secure Shell server host you are connecting to.

- **Destination port**: The destination port defines the port that is used for the forwarded connection on the destination host.

To edit a tunnel definition, select a tunnel from the list and click **Edit**. The **Local Tunnel** dialog opens.

To delete a tunnel definition, select a tunnel from the list and click **Delete** to remove a tunnel. Note that the selected tunnel will be removed immediately, with no confirmation dialog.

For more information on local tunnels, see Section 6.1.

Remote Tunnels

Remote (incoming) tunnels protect TCP connections that the remote host forwards from a specified remote port to the specified port on your local computer.

Click the **Remote tunnels** tab to edit incoming tunnel definitions. Click **Add** to open the **Remote Tunnel** dialog box.

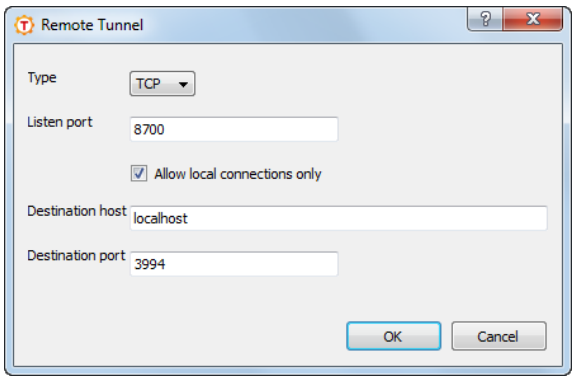


Figure A.26. Defining a remote tunnel

The following fields are used to define a remote tunnel:

- **Type**: Select the type of the tunnel from the drop-down list. Valid choices are TCP and FTP. For more information on FTP tunneling, see Section 6.1.3.
- **Listen port**: Enter the port that the tunnel listens to or captures from the remote host computer.

Note

Privileged ports (below 1024) can be forwarded only when logging in with root privileges on the remote host computer.

- **Destination host**: Define the destination host for the port forwarding. The default value is localhost.

Note

Here localhost refers to your local computer. Also note that if the connection from the remote host computer is forwarded beyond your local computer, that connection is unsecured.

- **Destination port**: Define the port that is used for the forwarded connection on the destination host.

To edit a tunnel definition, select a tunnel from the list and click **Edit**. The **Remote Tunnel** dialog opens.

To delete a tunnel definition, select a tunnel from the list and click **Delete** to remove a tunnel. Note that the selected tunnel will be removed immediately, with no confirmation dialog.

For more information on remote tunnels, see Section 6.2.

A.1.4 Defining User Authentication

Under **User Authentication**, you can configure settings related to public-key and certificate authentication. See the section called “Managing Keys and Certificates” and the section called “Managing Key Providers”.

To enable or disable public-key authentication, see the section called “Defining Default Connection Settings” and the section called “Defining Authentication”.

Managing Keys and Certificates

On the **Keys and Certificates** page, you can add key and certificate files used in user authentication and directories for them, generate a new key, upload a key to a server, or change the passphrase for a key.

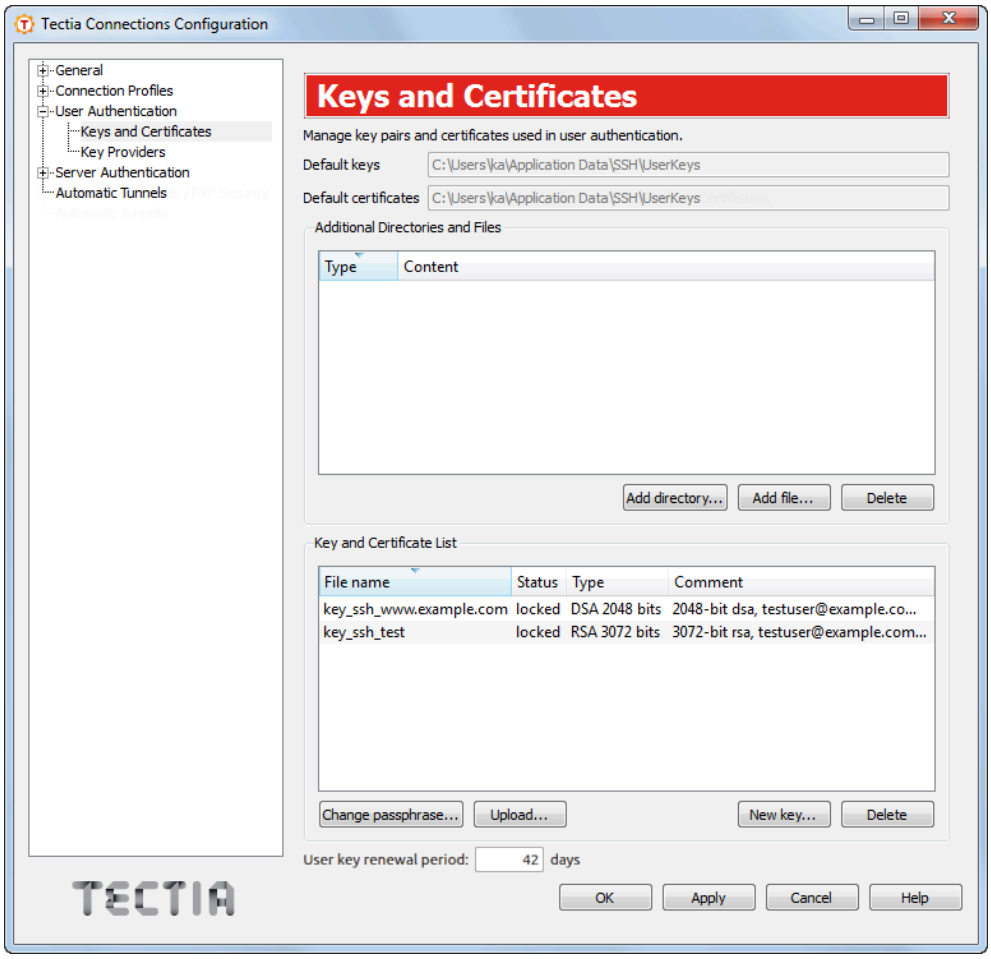


Figure A.27. Defining keys and certificates

Default keys

The default location of user keys.

Default certificates

The default location of user certificates.

Additional Directories and Files

Additional key directories and files explicitly added to the Tectia Client configuration.

- Click the **Add directory** button to add a directory of keys or certificates.
- Click the **Add file** button to add a key or certificate file.
- Select a directory or a file and click the **Delete** button to remove it. The reference to the directory, the key or certificate file is removed from the configuration. The keys themselves are not removed from the disk.

Key and Certificate List

All public keys and certificates known to Tectia Client are listed in this field. That is, those keys and certificates stored in locations in **Default keys**, **Default certificates** and **Additional Directories and Files** fields. Also the keys and certificates from external key providers are shown here (see the section called “Managing Key Providers”.

The value shown in the **Status** field can be:

- **locked** - The file is passphrase protected and the passphrase is not known to the Connection Broker. Uploading the file to a remote host unlocks it.
- **open** - The passphrase is known to the Connection Broker.
- If the field is empty, the file is not passphrase protected.

You can modify the key details by selecting a key file in the list and clicking a button at the bottom.

Click **Change passphrase** to change the passphrase of a selected key. Note that the command may not be supported for all key types.

Click **Upload** to upload the key to a remote server. You can only upload plain public keys. See also the section called “Uploading Public Keys Automatically”.

Click **New key** to start the key generation wizard. The new key will be added to the **Default keys** directory and it will become visible in the **Key and Certificate List** field. For a description of the wizard, see the section called “Using the Public-Key Authentication Wizard”.

Note

The user-specific `Application Data` directory, where the public key files are stored, is hidden by default. To view hidden directories, change the setting in Windows Explorer. For example, select **Organize → Folder and search options** on the menu. On the **View** tab, under **Hidden files and folders**, select **Show hidden files, folders and drives**.

User key renewal period

Set how many days it takes for automatic key rotation to happen. This affects the user keys in both the default key location, as well as the locations defined as additional directories above. Seperate key files do not support key rotation. If rotation period is set to *0*, the automatic key rotation is disabled.

When connecting to a host, the client will attempt to replace any keys older than the key rotation period with newly generated keys. This will not work if the server does not allow users to upload keys.

Warning: If the same private key has been copied to multiple clients, replacing the public key from one of them will break the others.

Managing Key Providers

On the **Key Providers** page you can define the settings of external key providers used in user authentication. Available key providers are Microsoft Crypto API and PKCS #11.

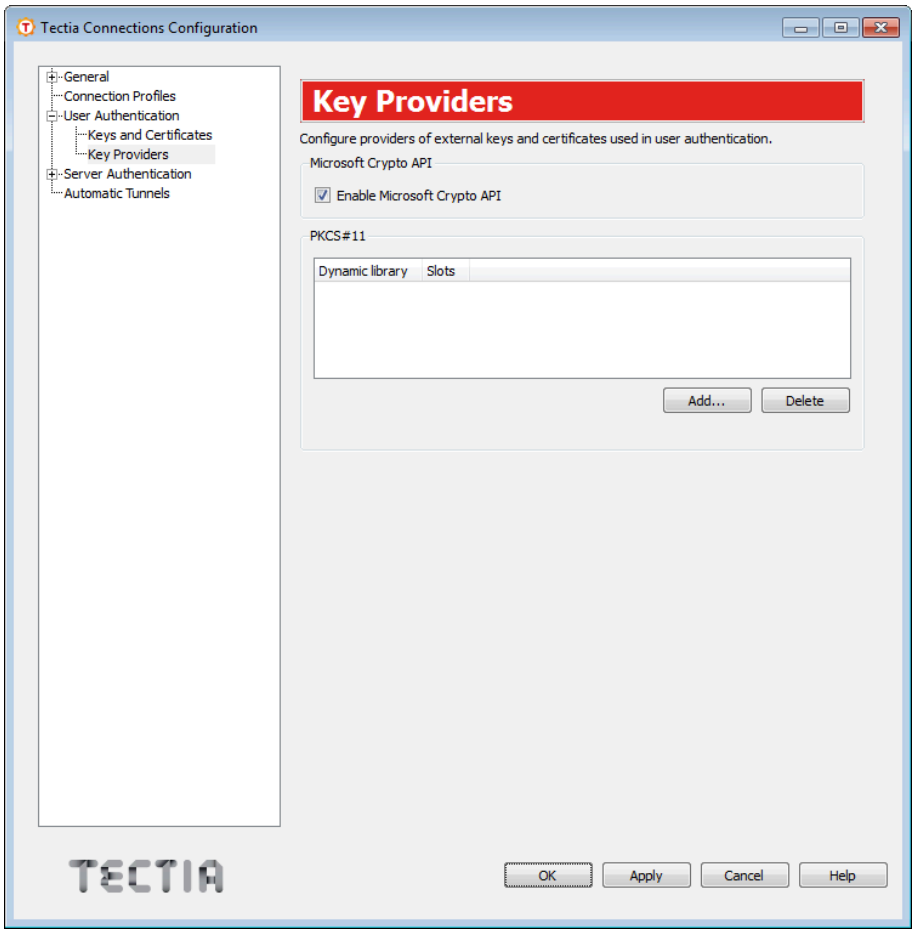


Figure A.28. Defining key providers

Microsoft Crypto API

Tectia Client can access keys via Microsoft Crypto API (MSCAPI). MSCAPI is a standard cryptographic interface used in Microsoft Windows systems.

Microsoft Crypto API (MSCAPI) providers can be enabled by selecting the **Enable Microsoft Crypto API** check box. If you enable the MSCAPI providers, you can use software keys and certificates created by Microsoft applications.

PKCS #11

By using the PKCS #11 provider, Tectia Client can use keys and certificates stored in PKCS #11 tokens (for example, smart cards or USB tokens).

Click **Add** to define a PKCS #11 provider.

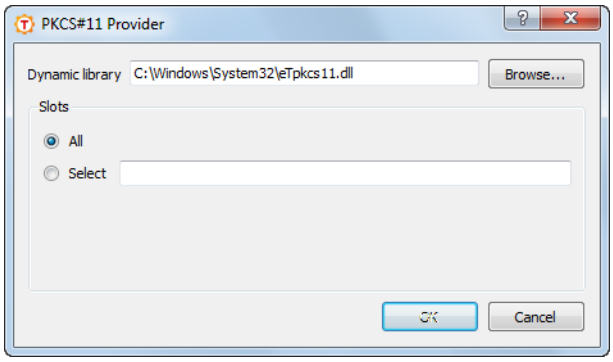


Figure A.29. Defining a PKCS #11 provider, Aladdin eToken DLL path shown as an example

Use the **Dynamic library** to define a dynamic library containing the PKCS #11 driver.

Use the **Slots** to define slots. A slot is a logical reader that potentially contains a token. Slots are manufacturer- specific. They are defined with an integer. Examples: "`0,1`", "`0-3, !2`", "`2`".

A.1.5 Defining Server Authentication

Under **Server Authentication**, you can define how Tectia Client authenticates remote server hosts.

- To use public keys in server authentication, define the settings as described in the section called "Managing Host Keys".
- To apply certificates, define the settings as described in the section called "Managing CA Certificates".
- Settings required for LDAP usage are described in the section called "Managing LDAP Server Settings".
- To define regular intervals for fetching certificate revocation lists (CRLs), see the section called "Managing CRL Prefetch Settings".

Managing Host Keys

On the **Host Keys** page, you can add new public host keys, define the host key acceptance policy, and view and manage known host keys used in server authentication. Known host keys mean keys already stored to the user-specific `%APPDATA%\SSH\Hostkeys` directory.

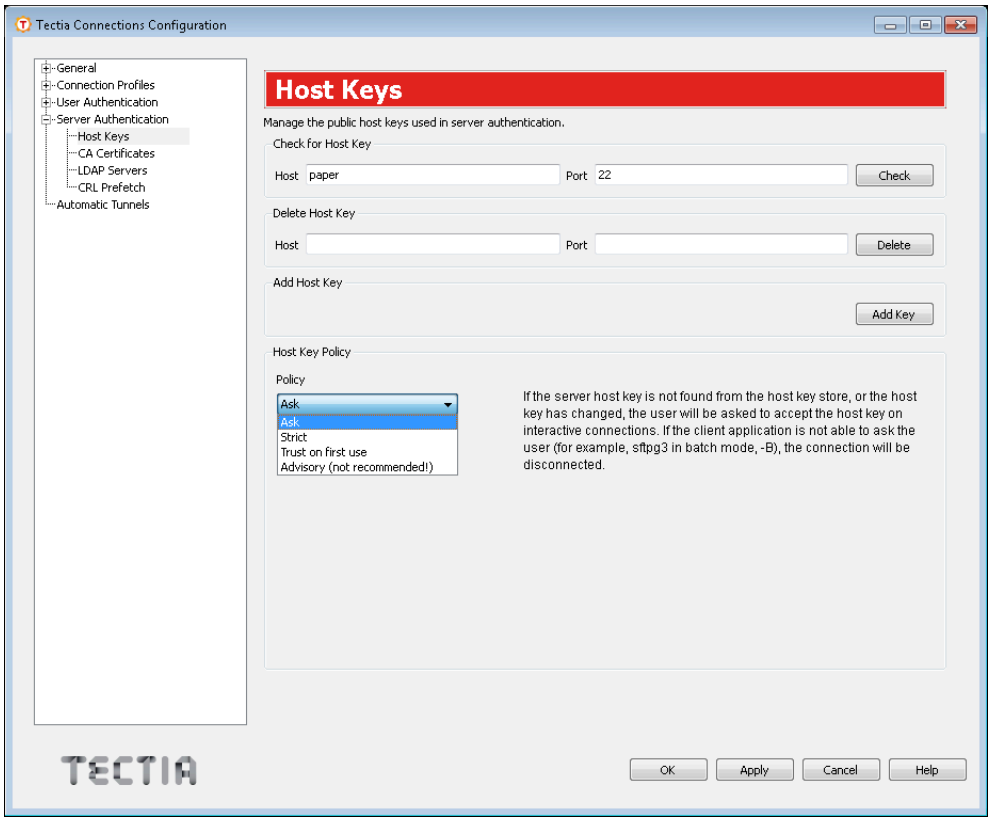


Figure A.30. Defining server host keys settings

Note

The host key policy settings have changed in version 6.1.4. Tectia Connections Configuration GUI updates the user-specific configuration automatically to use the new policy based on the old **Strict host key checking**, **Accept unknown host keys**, and **Always show host key prompt** settings. The interpretation of the old policy to the new policy is shown in Table A.2.

The **Host Keys** view includes the following options:

Check for Host Key

You can check if a public host key of a remote server exists on your client, and view its fingerprint. To check the host key, enter the name of the server in the **Host** field and the listener port number in the **Port** field, and click **Check**.

Note that wildcard characters are not allowed, specify the exact host name and port.

When a public host key for the specified server is found on the client, a dialog-box shows where the host key is stored and what is the fingerprint of the public key. The fingerprint is shown in the SSH Babble format, consisting of a series of pronounceable five-letter words in lower case and separated by dashes. See an example below.

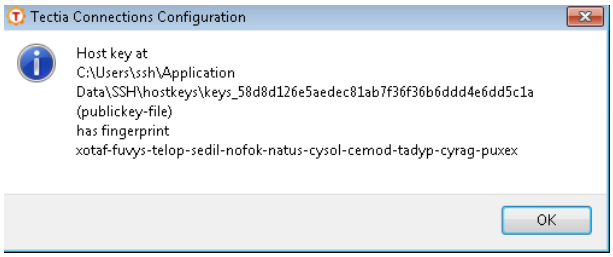


Figure A.31. Server public host key information

For more information on server public host keys, see Section 4.2.

Delete Host Key

In case you want to delete a known public host key from the client side, enter the name of the relevant server in the **Host** field and the listener port number in the **Port** field, and click **Delete**.

A dialog box appears asking you to confirm or to cancel the deleting of the host key.

Add Host Key

Click the **Add Key** button to add a new host key to your known host keys directory. The Connection Broker opens a file manager view where you can browse to the key location and select the host key you want to copy.

Host Key Policy

Select the policy you want to apply to the checking of server host keys and to the handling of unknown server host keys.

Note

This setting strongly affects the security of the client side host.

The options are:

- **Ask** - *the default* - the user will be asked to verify and accept the server public host keys, if the keys are not found in the host key store or if the keys have changed. The user can decide whether the key is to be stored to `%APPDATA%\SSH\Hostkeys`, or used once without storing it, or cancelled. Connection is allowed only to a server whose host key is either found in the known host keys directory or accepted by the user currently.

  This policy requires an interactive connection to get a response from the user. If the **Ask** option is applied on a non-interactive connection, the connection will be closed.

- **Strict** - the connection to the server will be allowed only if the host key is found in the user's known host keys storage. Otherwise, the connection will be closed. This option expects that all acceptable server host keys have already been stored on the client. No new host key's will be stored, and connections to any servers that have changed host keys will be closed.

This option can be used on non-interactive connections, once the host keys have been received by other means. This policy provides maximum protection against man-in-the-middle attacks.

- **Trust on first use** - new host keys are stored without prompting the user to accept them. Connections to servers offering a changed host key will be closed. This policy should be used only when server host keys cannot be added to the key storage by any other means.

- **Advisory** - *not recommended* - new host keys are stored without prompting the user to accept them, and connections are allowed also to servers offering a changed host key. Changed keys are not stored on the client, and data about opening connections with them are logged, provided that logging is enabled on the Connection Broker.

  If you choose this policy, make sure the Connection Broker has logging activated in the **General - Logging** view, see the section called “Defining Logging Settings”. Then you have the possibility to detect any connections with changed host keys in the logs.

Caution

Consider carefully before you activate the **Advisory** policy, as it practically disables server authentication and makes the connection vulnerable to active attackers.

Rotation

Select the rotation options you want to apply to the server host keys.

The options are:

- **No** - Disables key rotation.

- **Yes** - Enables key rotation.

- **Append only** - Enables key rotation. When this option is selected, the new key file is appended to the keyfile, without the old keys being removed.

- **Tectia only** - *the default* - Enables key rotation, but only for Tectia servers. This option requires enabling on the server also.

Managing CA Certificates

On the **Certificates** page, you can manage trusted CA certificates.

For more information on server certificate authentication, see Section 4.3.

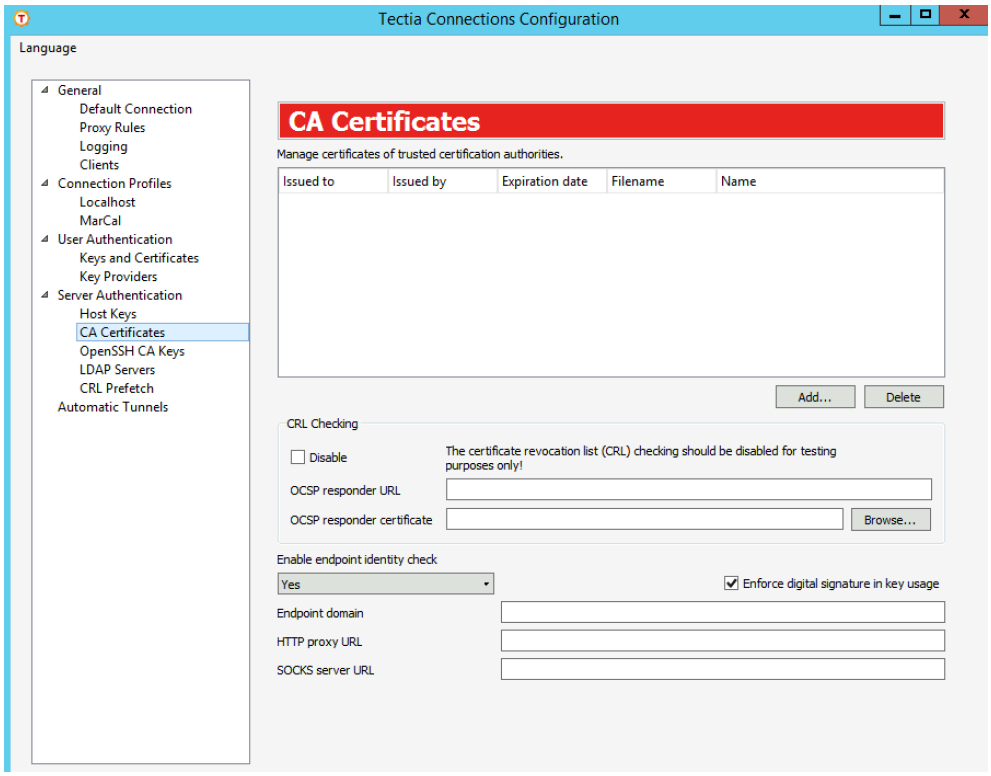


Figure A.32. Defining CA certificates

To add a CA certificate, click the **Add** button and select the certificate you want to add.

You can add X.509 certificate(s) as such. To add certificates from a PKCS #7 package (`.p7b`), you must first extract the CA certificates from the package by specifying the `-7` option with **ssh-keygen-g3** on the command line:

```
> ssh-keygen-g3 -7 certfile.p7b
```

You can then add the extracted CA certificates.

The following fields are displayed on the CA certificate list:

- **Issued to**: The certification authority to whom the certificate has been issued.
- **Issued by**: The entity who has issued the CA certificate.
- **Expiration date**: The date that the CA certificate will expire.
- **Filename**: The file containing the CA certificate.

CRL Checking

Select the **Disable** check box to prevent the use of a certificate revocation list (CRL). A CRL is used to check if any of the used server certificates have been revoked.

Note

Disabling CRL checking is a security risk and should be done for testing purposes only.

OCSP responder URL

The OCSP Responder Service provides client applications a point of control for retrieving real-time information on the validity status of certificates using the Online Certificate Status Protocol (OCSP).

For the OCSP validation to succeed, both the end-entity (=Secure Shell server) certificate and the OCSP responder certificate must be issued by the same CA. 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.

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.

Enable endpoint identity check

Specifies whether the client will verify the server's hostname or IP address against the **Subject Name** or **Subject Alternative Name** (DNS Address) specified in the server host certificate. By default, **Enable endpoint identity check** is enabled (option `yes`). The other options are `no`, and `ask`.

If `No` is selected, the fields in the server host certificate are not verified and the certificate is accepted based on the validity period and CRL check only.

Caution

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

If `ask` is selected, the user will be prompted to verify the certificate information and to either accept or cancel the connection.

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.

Endpoint domain

Specify the default domain used in the end-point identity check. This is the default domain part of the remote system name and it is used if only the base part of the system name is available.

If the default domain is not specified, the end-point identity check will still work with short host names. For example, when a user tries to connect to a host `"rock"` giving only the short host name and the certificate contains the full DNS address `"rock.example.com"`, the connection will be opened and Tectia Client will issue a warning about accepting a connection to `"rock"`.

HTTP proxy URL

Specify the HTTP proxy used when making LDAP or OCSP queries for certificate validity.

The format of the address is `"http://username@proxy_server:port/network/netmask,network/netmask... "`. The `network/netmask` part is optional and defines the network(s) that are connected directly (without the proxy).

SOCKS server URL

Specify the SOCKS server used when making LDAP or OCSP queries for certificate validity.

The format of the address is `"socks://username@socks_server:port/network/netmask,network/netmask... "`. The `network/netmask` part is optional and defines the network(s) that are connected directly (without the SOCKS server).

Managing OpenSSH CA Keys

On the OpenSSH CA Keys page, you can manage OpenSSH certificates.

To add an OpenSSH certificate, click the **Add** button.

To delete an OpenSSH certificate, select the certificate from the list, and click **Delete**.

Managing LDAP Server Settings

On the **LDAP Servers** page, you can define LDAP servers used for fetching CRLs and/or subordinate CA certificates based on the issuer name of the certificate being validated.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be verified if the point exists.

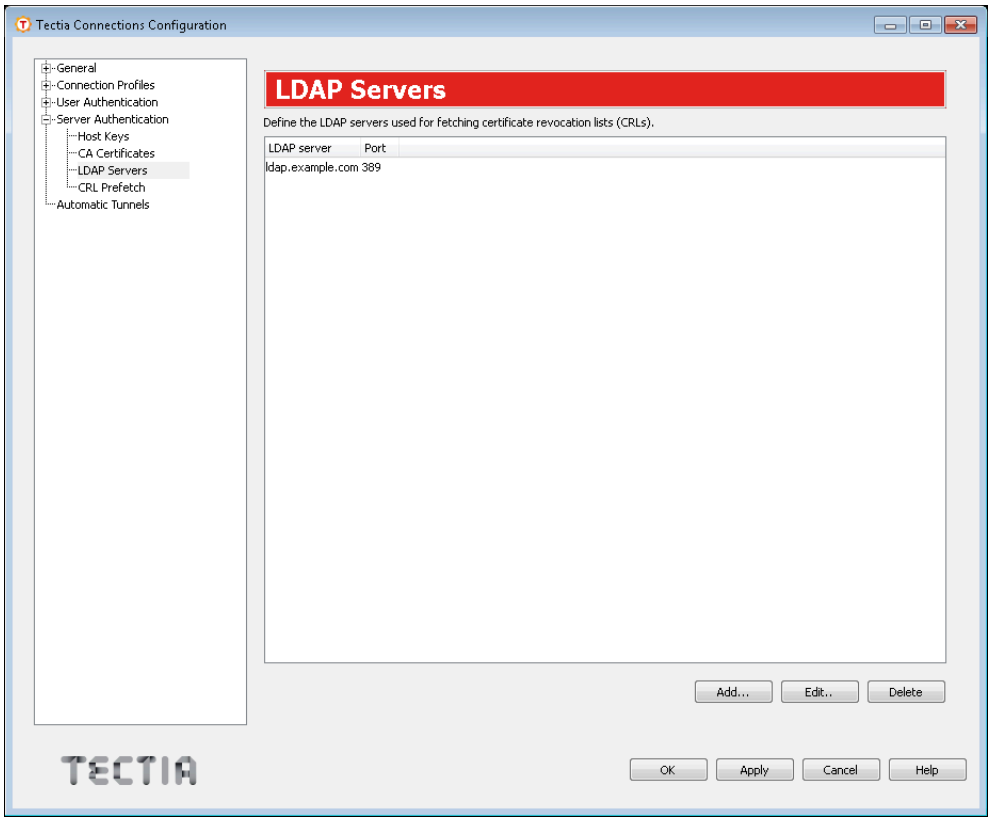


Figure A.33. Defining LDAP servers

To add an LDAP server, click the **Add** button. Define the hostname and port for the server.

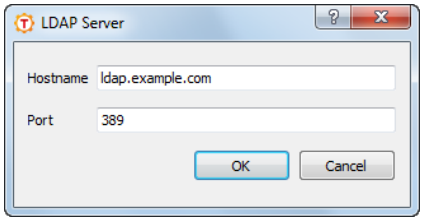


Figure A.34. Adding an LDAP server

To edit an LDAP server, select the server from the list and click **Edit**.

To delete an LDAP server, select the server from the list and click **Delete**.

Managing CRL Prefetch Settings

On the **CRL Prefetch** page, you can define certificate revocation lists (CRLs) to be fetched from the defined location at regular intervals. The CRL distribution point can be either a standard format LDAP or HTTP URL, or it can refer to a file. The file format must be either binary DER or base64, PEM is not supported.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be verified if the point exists.

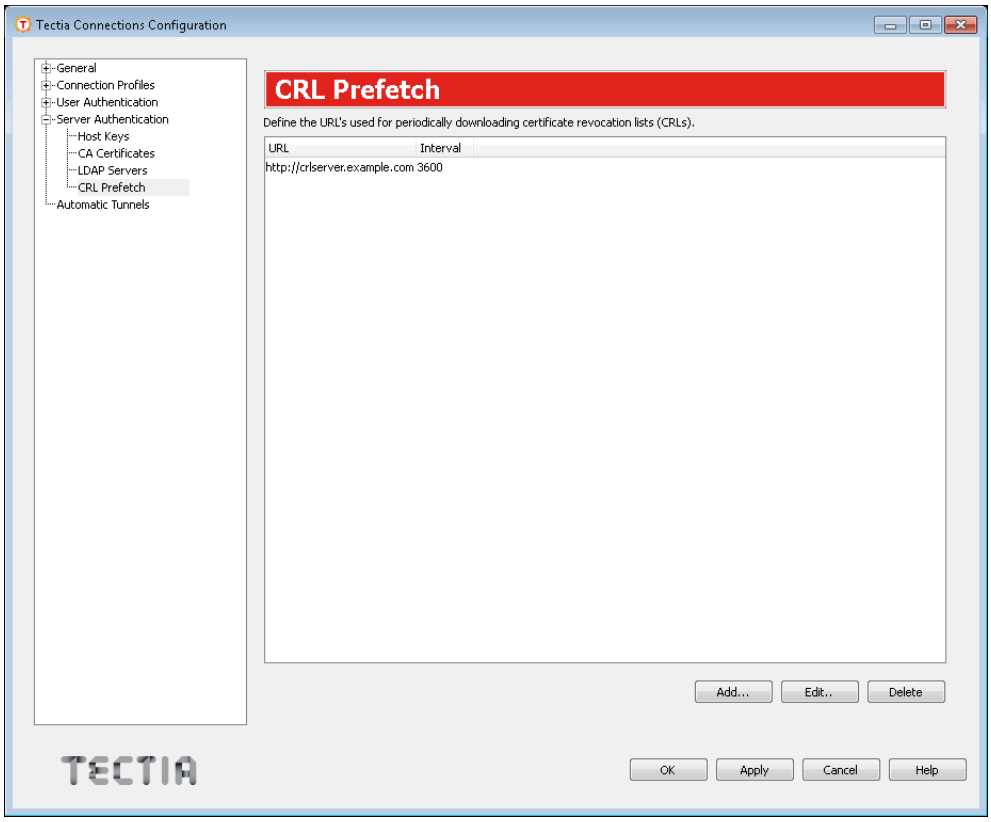


Figure A.35. Defining CRL prefetch settings

To add a CRL prefetch address, click **Add**. The **CRL Prefetch** dialog box opens.

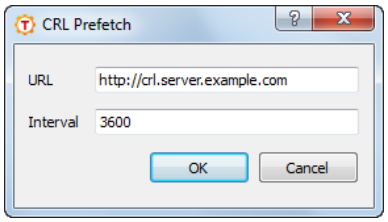


Figure A.36. Adding a CRL prefetch setting

Enter the **URL** of the CRL distribution point and the **Interval** how often the CRL is downloaded and click **OK**. The default download interval is `3600` (seconds).

In case the CRL distribution point refers to a file, enter the file URL in this format:

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

To edit an existing CRL prefetch setting, select the setting from the list and click **Edit**.

To delete an existing CRL prefetch setting, select the setting from the list and click **Delete**.

A.1.6 Defining Automatic Tunnels

On the **Automatic Tunnels** page, you can create listeners for local tunnels that are started automatically when the Connection Broker starts up. The actual tunnel is formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.

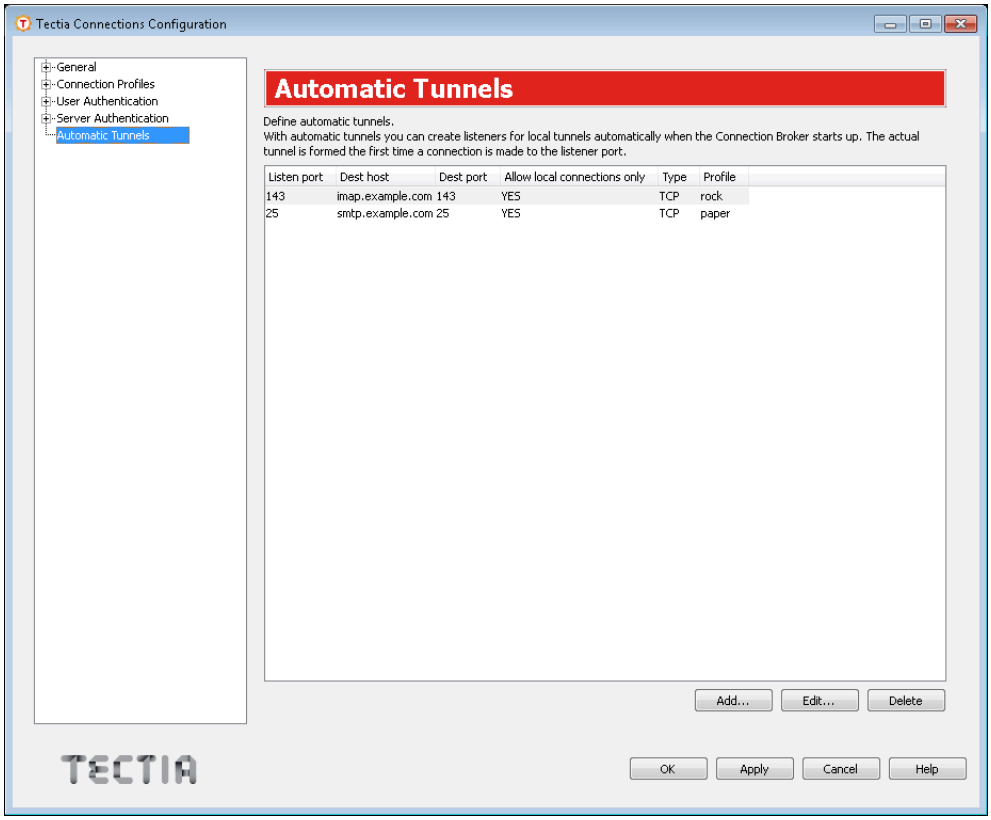


Figure A.37. Defining automatic tunnels

When the Connection Broker starts, the list of the automatic tunnels is read, and the connection initiating applications will be matched to the rules defined here.

Select **Automatic Tunnels** in the tree menu and click **Add** to open the **Automatic Tunnel** dialog box.

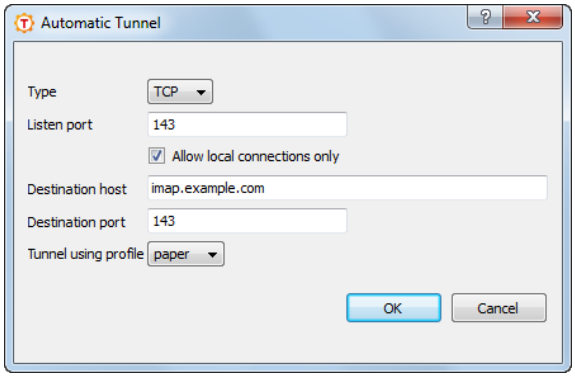


Figure A.38. Adding a new automatic tunnel

- **Type**: Select the type of the tunnel from the drop-down list. Valid choices are TCP and FTP.
- **Listen port**: This is the number of the local port that the tunnel listens to, or captures. Do not use a reserved port number.

Note

The protocol or application that you wish to create the tunnel for may have a fixed port number (for example 143 for IMAP) that it needs to use to connect successfully. Other protocols or applications may require an offset (for example 5900 for VNC) that you will have to take into account.

- **Allow local connections only**: If you want to allow only local connections to be made, leave this check box selected. This means that other computers will not be able to use the tunnel created by you. By default, only local connections are allowed. This is the right choice for most situations. You should carefully consider the security implications if you decide to also allow outside connections.
- **Destination host**: This field defines the destination host for the port forwarding. The default value is localhost.

Note

The value of localhost is resolved by the Secure Shell server, so here `localhost` refers to the Secure Shell host you are connecting to.

- **Destination port**: The destination port defines the port that is used for the forwarded connection on the destination host.
- **Tunnel using profile**: Select the profile to use for the tunnel.

To edit an automatic tunnel, select a tunnel from the list and click **Edit**.

To delete an automatic tunnel, select a tunnel from the list and click **Delete**.

For more information on tunneling, see Section 6.1.

A.2 Configuration File for the Connection Broker

The elements of the XML-based Connection Broker configuration file `ssh-broker-config.xml` are described in ssh-broker-config(5).

ssh-broker-config

ssh-broker-config — Tectia Connection Broker configuration file format

The Connection Broker configuration file `ssh-broker-config.xml` is used by Tectia Client on Unix and Windows. The Connection Broker configuration file must be a valid XML file that follows the `ssh-broker-ng-config-1.dtd` document type definition.

Connection Broker Files

The Connection Broker reads three configuration files (if all are available):

1. The `ssh-broker-config-default.xml` file is read first. It holds the factory default settings. It is not recommended to edit the file, but you can use it to view the default settings.

   This file must be available and correctly formatted for the Connection Broker to start.

2. Next, the Connection Broker reads the global configuration file. The settings in the global configuration file override the default settings.

   If the global configuration file is missing or malformed, the Connection Broker will start normally, and will read the user-specific configuration file, instead. A malformed global configuration file is ignored and the default settings or user-specific settings, if they exist, are used instead.

3. Last, the Connection Broker reads the user-specific configuration file, if it is available. The settings in the user-specific configuration file override the settings in the global configuration file, with the following exceptions:

   - The following settings from the user-specific configuration are combined with the settings of the global configuration file:
     - In `general` element, the `key-stores`, `cert-validation` and `file-access-control` settings
     - In `profiles` element, all settings
     - In `static-tunnels` element, all settings.
   - If a connection profile with the same name has been defined in both the global configuration file and user-specific configuration file, the latter one is used.
   - If the `filter-engine` settings have been defined in the global configuration file, and the file is valid (not malformed), those settings are used, and any `filter-engine` settings made in the user-specific configuration file are ignored.

   If the user-specific configuration file is missing, the Connection Broker will start using the previously read configuration files. However, if a user-specific configuration exists but is malformed, the Connection Broker will not start at all.

On Unix, the default configuration file locations are as follows:

- the default configuration:

  `/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`

- the global configuration: `/etc/ssh2/ssh-broker-config.xml`
- the user-specific configuration: `$HOME/.ssh2/ssh-broker-config.xml`
- the XML DTD:

  `/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd`

Note

In Tectia Client 6.1 and earlier on Unix the default auxiliary data directory `auxdata` was located in `/etc/ssh2/ssh-tectia/`. If your `ssh-broker-config.xml` file was created for Tectia Client version 6.1 or earlier, please update its DOCTYPE declaration to contain the current path to the Connection Broker configuration file DTD directory: `/opt/tectia/share/auxdata/ssh-broker-ng/`.

On Windows, the default configuration file locations are as follows (where `<INSTALLDIR>` indicates the default Tectia installation directory on Windows, see Section 1.1.2):

- Default configuration: "`<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml`"
- Global configuration: "`<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml`"
- User-specific configuration: "`%APPDATA%\SSH\ssh-broker-config.xml`"
- XML DTD: "`<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-ng-config-1.dtd`"

The following sections describe the options available in the Connection Broker configuration file. For more information on the syntax of the configuration file, see the XML DTD.

Environment Variables

Two kinds of environment variables can be used in the Connection Broker configuration file. In addition to the system-level environment variables, you can use special variables that are Tectia specific. The environment variables take precedence over the special variables. So if an environment variable and a special variable have the same name, the environment variable will be used.

All alphanumeric characters and the underscore '`_`' sign are allowed in environment variables. The variable name ends to the first character that is not allowed.

You can define for example file or directory paths with environment variables, and they will be expanded to their values as explained below.

%VARIABLENAME%

Replaced with the value of the environment variable if one has been defined. The variable is matched case-insensitively. If the variable is not defined, the string '`%VARIABLENAME%`' is the result.

$VARIABLENAME

Replaced with the value of the environment variable if one has been defined. The variable is matched case-sensitively on Unix and case-insensitively on Windows. If the variable is not defined, it is replaced with an empty string.

${VARIABLENAME}*text*

Replaced with the value defined for '`$VARIABLENAME`' with the '*text*' appended to it.

${VARIABLENAME:-*default_value*}

Replaced with the value defined for '`$VARIABLENAME`', or replaced with the '*default_value*' if the variable is not set.

The Tectia specific special variables are:

%U or %username%

Replaced with the currently logged in user name.

%username-without-domain%

Replaced with the currently logged in user name in short format, i.e. without the domain part. Available on Windows.

%G or %groupname%

Replaced with the group name of the currently logged in user.

%D or %homedir%

Replaced with the home directory defined for the currently logged in user.

%IU or %userid%

Replaced with the user identifier defined for the currently logged in user.

%IG or %groupid%

Replaced with the group identifier defined for the currently logged in user.

%installdir%

Replaced with the installation directory.

The special variables can also be entered using the Unix format, for example, `$username`.

Document Type Declaration and the Root Element

The Connection Broker configuration file is a valid XML file and starts with the Document Type Declaration.

The root element in the configuration file is `secsh-broker`. It can include `general`, `default-settings`, `profiles`, `static-tunnels`, `gui`, and `logging` elements.

An example of an empty configuration file is shown below:

```
<!DOCTYPE secsh-broker SYSTEM "ssh-broker-ng-config-1.dtd">
<secsh-broker version="1.0">
  <general />
  <default-settings />
  <profiles />
  <static-tunnels />
  <gui />
  <logging />

</secsh-broker>
```

The `general` Element

The `general` element contains settings such as the cryptographic library and the key stores to be used.

The `general` element can contain zero or one instance of the following elements: `crypto-lib`, `cert-validation`, `key-stores`, `user-config-directory`, `protocol-parameters`; and multiple `known-hosts` elements.

crypto-lib

This element selects the cryptographic library mode to be used. Either the standard version (`standard`) or the FIPS 140-2 certified version (`fips`) of the cryptographic library can be used. The library name is given as a value of the `mode` attribute. By default, standard cryptographic libraries are used. The OpenSSL cryptographic library is used in the *FIPS mode*.

FIPS mode will be used if it is so specified either in the global or the user configuration file (or both).

```
<crypto-lib mode="fips" />
```

For a list of platforms on which the FIPS library has been validated or tested, see *Tectia Client/Server Product Description.*

cert-validation

This element defines public-key infrastructure (PKI) settings user for validating remote server authentication X.509v3 certificates or OpenSSH CA-key for validating OpenSSH format server authentication certificates. The element can have the following attributes: `end-point-identity-check`, `default-domain`, `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 `end-point-identity-check` attribute specifies whether the client will verify the server's host name or IP address against the Subject Name or Subject Alternative Name (DNS Address) specified in the server host certificate. The default value is `yes`. If set to `no`, the fields in the server host certificate are *not* verified and the certificate is accepted based on the validity period and CRL check only.

Caution

Setting `end-point-identity-check="no"` is a security risk. Then anyone with a certificate issued by the same trusted certification authority (CA) that issues the server host certificates can perform a man-in-the-middle attack on the server.

Alternatively, if set to `ask`, the user can decide to either cancel or continue establishing the connection in case that the server's host name does not match the one in the certificate.

The `default-domain` attribute can be used when the end-point identity check is enabled. It specifies the default domain part of the remote system name and it is used if only the base part of the system name is available. The `default-domain` is appended to the system name if it does not contain a dot (`.`).

If the default domain is not specified, the end-point identity check will still work with short host names. For example, when a user tries to connect to a host "`rock`" giving only the short host name and the certificate contains the full DNS address "`rock.example.com`", the connection will be opened and Tectia Client will issue a warning about accepting a connection to "`rock`".

The `http-proxy-url` attribute defines an HTTP proxy and the `socks-server-url` attribute defines a SOCKS server for making LDAP or OCSP queries for certificate validity.

The address of the server is given as the value of the attribute. The format of the address is `socks://username@socks_server:port/network/netmask,network/netmask ...` (with a SOCKS server) or `http://username@proxy_server:port/network/netmask,network/netmask ...` (with an HTTP proxy).

For example, to make the SOCKS server use host `socks.ssh.com` and port `1080` for connections outside of networks `192.196.0.0` (16-bit domain) and `10.100.23.0` (8-bit domain), and to get these networks connected directly, set `socks-server-url` as follows:

```
"socks://mylogin@socks.ssh.com:1080/192.196.0.0/16,10.100.23.0/24"
```

The `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 multiple `ldap-server`, `ocsp-responder`, `crl-prefetch` elements, one `dod-pki` element, and multiple `ca-certificate`, `openssh-ca-key` and `key-store` elements. The elements have to be in the listed order.

ldap-server

This element specifies an LDAP server `address` and `port` used for fetching CRLs and/or subordinate CA certificates based on the issuer name of the certificate being validated. Several LDAP servers can be specified by using several `ldap-server` elements.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be verified if the point exists.

The default value for `port` is `389`.

ocsp-responder

This element specifies an OCSP (Online Certificate Status Protocol) responder service address in URL format with attribute `url`. Several OCSP responders can be specified by using several `ocsp-responder` elements.

If the certificate has a valid Authority Info Access extension with an OCSP Responder URL, it will be used instead of this setting. Note that for the OCSP validation to succeed, both the end-entity certificate and the OCSP Responder certificate must be issued by the same CA.

The `validity-period` (in seconds) can be optionally defined. During this time, new OCSP queries for the same certificate are not made but the old result is used. The default validity period is `0` (a new query is made every time).

crl-prefetch

This element instructs Tectia Client to periodically download a CRL from the specified URL. The `url` value can be an LDAP or HTTP URL, or it can refer to a local file. The file format must be either binary DER or base64, PEM is not supported.

To download CRLs from the local file system, define the file URL in this format:

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

To download CRLs from an LDAP server, define the LDAP URL in this format:

```
ldap://ldap.server.com:389/CN=Root%20CA,
       OU=certification%20authorities,DC=company,
       DC=com?certificaterevocationlist
```

Use the `interval` attribute to specify how often the CRL is downloaded. The default is `3600` seconds.

dod-pki

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 defines a certification authority (CA) used in server authentication. It can have four attributes: `name`, `file`, `disable-crls`, and `use-expired-crls`.

The `name` attribute must contain the name of the CA.

The element must either contain the path to the X.509 CA certificate file as a value of the `file` attribute, or include the certificate as a base64-encoded ASCII block.

CRL checking can be disabled by setting the `disable-crls` attribute to `yes`. The default is `no`.

Expired CRLs can be used by setting a numeric value (in seconds) for the `use-expired-crls` attribute. The default is `0` (do not use expired CRLs).

openssh-ca-key

This element defines an OpenSSH certification authority (CA) used in server authentication. It can have two attributes: `name` and `file`.

The `name` attribute 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 block.

Generic cert-validation elements except `end-point-identity-check` do not apply to OpenSSH certificate validation, as there are no revocation services.

An example of an OpenSSH certificate validation configuration:

```
<cert-validation end-point-identity-check="ask">
  <openssh-ca-key name="OpenSSH_CA"
                  file="openssh_ca_key.pub" />
</cert-validation>
```

key-store

This element defines CA certificates stored in an external key store for server authentication. Currently it is used only on z/OS for CA certificates stored in System Authorization Facility (SAF).

An example of a certificate validation configuration is shown below:

```
<cert-validation  end-point-identity-check="yes"
                  default-domain="example.com"
                  http-proxy-url="http://proxy.example.com:8080">
  <ldap-server    address="ldap://ldap.example.com:389" />
  <ocsp-responder url="http://ocsp.example.com:8090"
                  validity-period="0" />
  <crl-prefetch   url="file:///full.path.to.crlfile"
                  interval="1800" />
  <dod-pki        enable="no" />
  <ca-certificate name="ssh_ca1"
                  file="ssh_ca1.crt"
                  disable-crls="no"
                  use-expired-crls="100" />
</cert-validation>
```

key-stores

This element defines settings for user public-key and certificate authentication.

Under the `<general>` element, there can be one `<key-stores>` instance which in turn can have any number of `<key-store>`, `<user-keys>`, and `<identification>` elements, and the order of the elements is free.

Special variables and environment variables can be used when defining the values for the elements. The following variables can be used and they will be expanded as follows:

- `%U` = `%USERNAME%` = user name
- `%USERNAME-WITHOUT-DOMAIN%` = user name without the domain part
- `%IU` = `%USERID%` = user ID (not on Windows)
- `%IG` = `%GROUPID%` = user group ID (not on Windows)
- `%D` = `%HOMEDIR%` = the user's home directory
- `%G` = `%GROUPNAME%` = the name of the user's default group
- `%INSTALLDIR%` = the installation directory

Also environment variables are replaced with their current values. For example it is possible to use strings `$HOME` or `%HOME%` to expand to user's home directory (if environment variable `HOME` is set).

Note

Short alias names (for example, `%U`) are case-sensitive and long alias names (for example, `%USERNAME%`) are case-insensitive.

key-store

Each of the `key-store` elements configures one key store provider. The `key-stores/key-store` element can take the following attributes: `type` and `init`.

The `type` attribute is the key store type. The currently supported types are `"mscapi"`, `"pkcs11"`, `"software"`, and `"zos-saf"`.

The `init` attribute is the initialization info specific to the key-store-provider. The initialization string can contain special strings explained above in **key-stores** .

For key store configuration examples, see the section called “Key Store Configuration Examples”.

user-keys

The `user-keys` element can be used to override the default directory for the user keys. The `user-keys` element can take the following attributes:

The `directory` attribute defines the directory where the user private keys are stored. Enter the full path.

The `passphrase-timeout` attribute defines the time (in seconds) after which the passphrase-protected private key will time out, and the user must enter the passphrase again. The default is `0`, meaning that the passphrase does not time out. The value of this element should be longer than the `passphrase-idle-timeout` value.

By default, the Connection Broker keeps the passphrase-protected private keys open once the user has entered the passphrase successfully. This can be changed with the passphrase timeout options. When `passphrase-timeout` is set, the private key stays open (usable without further passphrase prompts) until the timeout expires. The `passphrase-timeout` attribute sets the hard timeout, that is set only once when the key is opened and will not be reset even if the key is used multiple times.

The `passphrase-idle-timeout` attribute defines the time (in seconds) after which the passphrase-protected private key will time out unless the user accesses or uses the key. The `passphrase-idle-timeout` is reset every time the key is accessed. The default is `0`, meaning that the passphrase never times out.

Both of the timeout options can be set simultaneously, but notice that if the idle timeout is set longer than the hard timeout, the idle timeout has no effect.

The `rotation-period` attribute defines the time (in seconds) after which the key will be rotated. Note that you can use the suffixes `m`, `minutes`, `h`, `hours`, `d` and `days` to define the time period.

identification

The `identification` element can be used to override the default location of the identification file that defines the user keys. The `identification` element can take the following attributes:

The `file` attribute specifies the location of the identification file. Enter the full path.

The `base-path` attribute defines the directory where the identification file expects the user private keys to be stored. This element can be used to override the default relative path interpretation of the identification file (paths relative to the identification file directory).

The `passphrase-timeout` attribute defines the time (in seconds) after which the user must enter the passphrase again. The default is `0`, meaning that the passphrase is not re-requested.

The `passphrase-idle-timeout` attribute defines a time (in seconds) after which the passphrase times out if there are no user actions. The default is `0`, meaning that the passphrase does not time out.

The timeout settings affect only those private keys that are listed in the identification file.

strict-host-key-checking

Note

This element is deprecated starting from Tectia Client version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `host-key-always-ask` and `accept-unknown-host-keys` options. See **auth-server-publickey** for details.

host-key-always-ask

Note

This element is deprecated starting from Tectia Client version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `strict-host-key-checking` and `accept-unknown-host-keys` options. See **auth-server-publickey** for details.

accept-unknown-host-keys

Note

This element is deprecated starting from Tectia Client version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `strict-host-key-checking` and `host-key-always-ask` options. See **auth-server-publickey** for details.

Caution

Consider carefully before enabling this option. Disabling the host-key checks makes you vulnerable to man-in-the-middle attacks.

user-config-directory

This element can be used to change the storage location of the user-specific configuration files away from the default which is `$HOME/.ssh2/` on Unix, and "`%APPDATA%\SSH`" on Windows. It can be used for example, if you want to store all client-side configurations to a centralized location.

When this element is added to the global configuration file, the Connection Broker reads the following user-specific files in the defined location:

- User's key file
- User's own configuration files
- User's known host keys
- User's random_seed file
- Windows GUI profile files: `1.ssh2, 2.ssh2`
- The startup batch file for the **sftpg3** client: `ssh_sftp_batch_file`

Note

Stop all existing SSH applications before modifying the `user-config-directory` setting in the Connection Broker configuration.

The `user-config-directory` setting affects all Tectia products running on the same host.

The `user-config-directory` option takes an attribute `path`, whose value can be either a directory path or one of the following variables:

- `%U`: The user name.
- `%username%`: The user name.
- `%username-without-domain%`: The user name without domain definition.
- `%D`: The user's home directory.
- `%homedir%`: The user's home directory.
- `%USER_CONFIG_DIRECTORY%`: The user-specific configuration directory.
- `%installdir%`: The installation directory.
- `%IU`: The user's ID, on Unix only
- `%userid%`: The user's ID, on Unix only

- `%IG`: The group ID, on Unix only
- `%groupid%`: The group ID, on Unix only

The default is `%USER_CONFIG_DIRECTORY%`. This variable refers to the user-specific configuration directory: `$HOME/.ssh2` on Unix, and `%APPDATA%\SSH` on Windows. The `%USER_CONFIG_DIRECTORY%` variable cannot be used in other settings.

file-access-control

On Unix, this element can be used to enable checking of file access permissions defined for the global and user-specific configuration files, and for the private keys files. If the permissions are not as expected, the Connection Broker will refuse to start, or to use certain private keys.

By default this setting is disabled. On Windows, this element has no effect.

The file permissions are checked differently, if the `file-access-control` element is set in both the global and user configuration files, or just in one of them. See the following table for details:

Table A.1. Different file-access-control effects

Setting in:		Permissions checked in:		
Global config	**User config**	**Global config**	**User config**	**Private key files**
yes	yes / -	checked	checked	checked
yes	no	checked	checked	not checked
no / -	yes	not checked	checked	checked
no / -	no / -	not checked	not checked	not checked

In the table: `"no"` means `file-access-control enable="no"`. The `"-"` sign means that the setting is not defined in the file at all.

When the file access permissions are checked, the controls are applied as follows:

- Expected permissions for the global configuration file: read rights for all, write rights only for the user and group. If the permissions are any wider, the Connection Broker will not start.
- Expected permissions for the user configuration file: only the user has read and write rights. If the permissions are any wider, the Connection Broker will not start.
- Expected permissions for the private key files: only the user has read and write rights. If the permissions are any wider, keys that do not pass the check will be ignored.

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" />
```

known-hosts

This element can be used to specify locations for storing the host keys of known server hosts, and to define the storage format of the host key files. If no `known-hosts` directories are specified, the known host keys are stored to the default directories. See the section called “Files” for the default locations. On z/OS (only), this element can contain `key-store` elements.

This element can be used:

- To specify non-default directories that contain the public-key data or public-key files of known server hosts.
- To specify a non-default location for OpenSSH-style known_hosts files that contain the public-key data of known server hosts.
- (*On z/OS*) To specify a SAF key store that contains the certificates of known server hosts.

The server host keys are searched in the `known-hosts` paths in the order they are specified in the configuration. The settings of the last defined `known-hosts` element are used when storing new host keys.

If you define any `known-hosts` file settings, the default OpenSSH files will be overridden. So if you wish to make the Connection Broker use both the default OpenSSH locations and other locations specified in the configuration, you need to specify all the locations separately.

You can define several `known-hosts` elements, and each of them can contain one or several attributes: `path`, `directory`, `file` and `filename-format`.

The `path` attribute requires a full path to the known-hosts file or directory as the value. For example:

```
<known-hosts path="/u/username/.ssh/known_hosts" />
<known-hosts path="/etc/ssh2/hostkeys" />
<known-hosts path="/u/username/.ssh2/hostkeys" />
<known-hosts path="/h/username/hostkeys" filename-format="plain" />
```

The `directory` attribute is used to define that known host keys are saved to a non-default directory. Enter the complete path to the directory as the value. If the defined directory does not exist, it will be created during the first connection attempt. If a file is found in its place, the connection will be made but the host key will not be stored, and the user gets a warning about it. The `filename-format` attribute can be used together with the `directory` setting to define in which format the host key files will be stored. Example of the `directory` attribute:

```
<known-hosts directory="<path_to_dir>/MyKEYS"
             filename-format="plain" />
```

The `path` or `directory` (whichever is present) defined in the last `known-hosts` element in the configuration file will be used when storing new known host keys. If both attributes are present in the last `known-hosts` element, the location specified in the `directory` attribute will be used.

The `file` attribute is used to point to an OpenSSH-style known_hosts file. Enter the complete path to the file as the value. If a directory is found in its place, it is considered an error, and the connection attempt will fail. In case the `known-hosts` element only contains the `file` attribute, and the defined OpenSSH known_hosts file exists, the received host keys are searched first in the defined file, and if not found there, the search continues in the default Tectia-specific locations.

Example of the `file` attribute:

```
<known-hosts file="<path_to_file>/.ssh2/openSSH_keys" />
```

An empty `file` or `path` attribute will disable the handling of the OpenSSH known_hosts file:

```
<known-hosts file="" />
or
<known-hosts path="" />
```

The `filename-format` attribute defines the format in which new host key files are stored. The `filename-format` attribute is only relevant for the last specified `known-hosts` element and for the default directory.

The `filename-format` attribute takes the values: `hash` (default), `plain`, and `default` (equals to hash).

With value `hash`, the host key files will be stored in format: `keys_<hash>`, for example "`keys_182166d2efe5a134d3fb948646e0b48f780bff6c`".

With value `plain`, the file name format will be `key_<port>_<hostname>.pub`, where `<port>` is the port the Secure Shell server is running on and `<hostname>` is the host name you use when connecting to the server; for example "`key_22_my.example.com.pub`".

Setting `<known-hosts filename-format="plain" />` changes the storage format of host key files for the next `known-hosts` elements or for the default storage location if no other `known-hosts` elements are present.

The `filename-format="default"` alternative can be used as the last option when the same `known-hosts` element is used to define several locations for the host keys some of which store the keys in plain format.

For more information on the host key storage formats, see Section 4.2.1.

key-store

This element defines an external key store for certificates of known server hosts. Currently it is used only on z/OS for server certificates stored in System Authorization Facility (SAF).

extended

This element is reserved for future use.

Key Store Configuration Examples

Example with Software Provider

The software provider handles key pairs stored on disk in standard Secure Shell v2 or legacy OpenSSH formats and X.509 certificates stored in native X.509, PKCS #7, and PKCS #12 formats.

To add a single key file (for example, `/u/exa/keys/enigma` and `/etc/my_key`), specify both the private key file and the public key file:

```
<key-stores>
  <key-store type="software"
             init="key_files(/u/exa/keys/enigma.pub,/u/exa/keys/enigma)" />
  <key-store type="software"
             init="key_files(/etc/my_key.pub,/etc/my_key)" />
</key-stores>
```

To add all keys from a specific directory (for example all keys from `/u/exa/keys` and `/etc/keys`):

```
<key-stores>
  <key-store type="software"
             init="directory(path(/u/exa/keys))" />
  <key-store type="software"
             init="directory(path(/etc/keys))" />
</key-stores>
```

Example with PKCS #11 Provider

The PKCS #11 provider handles keys and certificates stored in PKCS #11 tokens (for example, smart cards or USB tokens).

Specify the dynamic library path for the PKCS provider and all or a specific slot. For example, with all slots:

```
<key-stores>
  <key-store type="pkcs11" init="dll(/usr/lib/pkcs.so),slots(all)" />
</key-stores>
```

For example, with one slot named `sesam`:

```
<key-stores>
  <key-store type="pkcs11" init="dll(/usr/local/lib/pkcs.so),slots(sesam)" />
</key-stores>
```

The `default-settings` Element

The `default-settings` element defines the default connection-related settings. Profile-specific settings can override these settings. See the section called “The `profiles` Element”.

The `default-settings` element can contain zero or one instance of the following elements in the listed order: `ciphers`, `macs`, `kexs`, `hostkey-algorithms`, `rekey`, `authentication-methods`, `hostbased-default-domain`, `compression`, `proxy`, `idle-timeout`, `tcp-connect-timeout`, `keepalive-interval`, `exclusive-connection`, `server-banners`, `forwards`, `extended`, `remote-environment`, `server-authentication-methods`, `authentication-success-message`, `sftpg3-mode`, `terminal-selection`, `terminal-bell`, `close-window-on-disconnect`, `quiet-mode`, `checksum`, and `address-family`.

The **default-settings** element can take one attribute:

The `user` attribute can be used to define a default user name to be used when connecting to remote servers. The value of the `user` attribute can be one of the following:

- A generic user name that will be used in connections unless another user name is specified in the connection profile settings or in the connection attempt. Note that the user name is treated case sensitively.
- "`%USERNAME%`" can be used to apply the user name of the currently logged in user.
- In case this option is used but left empty, the Connection Broker will prompt the user for a user name.

The **default-settings** element can contain the following elements:

ciphers

This element defines the ciphers that the client will propose to the server. The `ciphers` element can contain multiple `cipher` elements.

The ciphers are tried in the order they are specified.

cipher

This element selects a cipher `name` that the client requests for data encryption.

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

```
<ciphers>
  <cipher name="aes128-cbc" />
  <cipher name="AEAD_AES_256_GCM" />
</ciphers>
```

macs

This element defines the MACs that the client will propose to the server. The `macs` element can contain multiple `mac` elements.

The MACs are tried in the order they are specified.

mac

This element selects a MAC `name` that the client requests for data integrity verification.

The list of supported MAC algorithms can be found in Section E.3.

```
<macs>
  <mac name="hmac-sha2-512" />
</macs>
```

kexs

This element defines the key exchange methods (KEXs) that the client will propose to the server. The `kexs` element can contain multiple `kex` elements.

The KEXs are tried in the order they are specified.

kex

This element selects a KEX `name` that the client requests for the key exchange method.

The list of supported classical and PQC hybrid KEX methods can be found in Section E.2.

```
<kexs>
   <kex name="ecdh-nistp521-kyber1024-sha512@ssh.com" />
   <kex name="ecdh-sha2-nistp256" />
</kexs>
```

hostkey-algorithms

This element defines the host key signature algorithms used for server authentication. The algorithms that will be used are those that are defined in both Tectia Server and Connection Broker configuration files. This way the use of only certain algorithms, such as SHA-2, can be enforced by the server. The `hostkey-algorithms` element can contain multiple `hostkey-algorithm` elements.

The hostkey algorithms are tried in the order they are specified. Exception: If a host key of a server already exists in the host key store of the client, its algorithm is preferred.

hostkey-algorithm

This element selects a host key signature algorithm `name` to be used in server authentication with host keys or certificates.

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

```
<hostkey-algorithms>
   <hostkey-algorithm name="rsa-sha2-256" />
   <hostkey-algorithm name="ssh-dss-sha512@ssh.com" />
</hostkey-algorithms>
```

rekey

This element specifies the number of transferred `bytes` after which the key exchange is done again. The value `"0"` turns rekey requests off. This does not prevent the server from requesting rekeys, however. The default is 1000000000 (1 GB).

```
<rekey bytes="1000000000" />
```

authentication-methods

This element specifies the authentication methods that are requested by the client-side components. The `authentication-methods` element can contain one of each: `auth-hostbased`, `auth-password`, `auth-publickey`, `auth-gssapi`, and `auth-keyboard-interactive`. Alternatively, you can specify multiple `authentication-method` elements. The order of these elements is free.

The authentication methods are tried in the order the `auth-*` or `authentication-method` elements are listed. This means that the least interactive methods should be placed first.

When several interactive authentication methods are defined as allowed, Tectia Client will alternate between the methods and offers each of them in turn to the server in case the previous method failed.

authentication-method

This element specifies an authentication method `name`. It is included for backwards compatibility. Use the `auth-*` elements instead.

The allowed authentication method names are: `gssapi-with-mic`, `publickey`, `keyboard-interactive`, `password`, and `hostbased`.

Tectia Client supports host-based authentication only on Unix platforms.

```
<authentication-methods>
  <authentication-method name="hostbased" />
  <authentication-method name="gssapi-with-mic" />
  <authentication-method name="publickey" />
  <authentication-method name="keyboard-interactive" />
  <authentication-method name="password" />
</authentication-methods>
```

auth-hostbased

This element specifies that host-based authentication will be used.

The `auth-hostbased` element can include a `local-hostname` element.

local-hostname

This element specifies the local host name, as the value of the `name` attribute, that is advertised to the remote server during host-based authentication.

The remote server can use the client host name as a hint when locating the public key for the client host. This information is not significant to the authentication result, but makes it faster to find the relevant client host key, if the server has such a big storage of host identities, that trying them all would be infeasible.

auth-password

This element specifies that password authentication will be used.

auth-publickey

This element specifies that public-key authentication will be used.

The `auth-publickey` element can include a `key-selection` element.

The `auth-publickey` element can include a `signature-algorithms` attribute. The attribute defines the public-key signature algorithms used for client authentication, given as a comma-separated list. The algorithms that will be used are those that are defined in both Tectia Server and Connection Broker configuration files. This way the use of only certain algorithms, such as SHA-2, can be enforced. For a list of the supported algorithms, see **hostkey-algorithm** .

```
<authentication-methods>
  <auth-publickey signature-algorithms="ssh-ed25519,ssh-rsa-sha256@ssh.com"/>
</authentication-methods>
```

key-selection

This element specifies the key selection policy the client uses when proposing user public keys to the server. The `policy` attribute can take the values `automatic` (default) and `interactive-shy`.

In the `automatic` mode, the client tries keys in the following order:

1. Keys with public key available and private key without a passphrase (no user interaction)
2. Keys with public key available but private key behind a passphrase (one passphrase query)
3. Keys that need a passphrase to get the public key but private key without passphrase (one user query for each key which is considered and proposed to server, but no user interaction for actual public-key login)
4. The rest of the keys, that is, keys that need a passphrase to get the public key and also to get the private key

In the `interactive-shy` mode, the client does not try any keys automatically, but it prompts the user to select the key from a list of available keys. If the authentication with the selected key fails, the client will prompt the user again, removing the already tried key(s) from the list. If there is only one key candidate available, the key will be tried automatically without asking the user.

The `key-selection` element can include the following:

```
altname-email
altname-upn
extended-key-usage
issuer-name
public-key
subject-name
validity
```

Example key-selection element:

```
<key-selection>
  <issuer-name name="CN=issuingcaname" pattern=".*testca.*" />
  <subject-name name="CN=username" pattern=".*username.*" />
  <extended-key-usage oid="ssh-client" explicit="yes" />
  <extended-key-usage oid="1.3.6.1.4.1.4449.1.2.4.1.3" />
  <validity valid-for="3600" />
  <altname-email name="user@something.com" pattern="user@.*" />
  <altname-upn name="user@something.com" pattern="user@.*" />
</key-selection>
```

altname-email

Filter user certificates by email in subjectAltName. Use the `name` option for strict matching and/or `pattern` for regex matching.

Supports variable substitutions (described later).

altname-upn

Filter user certificates by UPN in subjectAltName. Use the `name` option for strict matching and/or `pattern` for regex matching.

Supports variable substitutions (described later).

extended-key-usage

Filter user certificates by usage extensions. Use the `oid` option to specify an extension in name or OID format. If `explicit` is set the extended key usage must be in the certificate. If `explicit` is not set then certificates without any extended key usage will be considered matching.

issuer-name

This element can be used to filter the user certificates that will be included in the list presented to the user. The client-side user certificates can be filtered according to the issuer name that is compared to the certificate issuers requested or accepted by the server. The `match-server-certificate` attribute takes values `yes` and `no`. With value `yes`, Connection Broker tries matching the user certificate issuer name to the server certificate issuer name. Option `no` means that the issuer names are not used as a filter. By default, the filtering is not done.

The `issuer-name` is useful when a user has several certificates with different access rights to the same server, for example for a testing role and for an administrator role. The Connection Broker chooses the relevant certificates that are applicable on the remote host, and the user can choose the correct certificate from the short-listed ones.

Supports variable substitutions (described later).

public-key

This element can be used to specify that only plain public keys or only certificates are tried during public-key authentication. The `type` attribute can take the values `plain` and `certificate`. The default is to try both plain public keys and certificates.

subject-name

Filter user certificates by subject name. Use the `name` option for strict matching and/or `pattern` for regex matching.

Supports variable substitutions (described later).

validity

Filter user certificates by remaining validity period. Only those certificates where remaining validity exceeds `valid-for` are matched. `valid-for` supports time units 's', 'm', 'h' and 'd' (seconds, minutes, hours, and days), for example `3d 18h`. When time unit is unspecified, the duration is assumed to be in seconds.

You may use variable substitutions in `altname-email`, `altname-upn`, `issuer-name`, and `subject-name` elements. Specify variable substitutions in *`%variable_name%`* format, where *`variable_name`* can be any of the following:

- username: The local user name.
- username-without-domain: The local user name without the domain part.
- userhome: User's home directory
- uid: The user's user ID. Only available on Linux.
- gid: The user's group ID. Only available on Linux.
- group-name: The user's group name. Only available on Linux.
- remote-user: User name specified in the SSH-connection command.
- remote-host: Host name specified in the SSH-connection command.
- remote-port: Port number specified in the SSH-connection command.
- peer-version: The SSH protocol version.
- session-id: The session ID in hex format.

auth-keyboard-interactive

This element specifies that keyboard-interactive methods will be used in authentication.

auth-gssapi

This element specifies that GSSAPI will be used in authentication.

The `auth-gssapi` element can take the following attributes:

The `dll-path` attribute 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 Client locates the correct DLL automatically.

The `allow-ticket-forwarding` attribute defines whether Tectia Client allows forwarding the Kerberos ticket over several connections. The attribute can have a value of `yes` or `no`. The default is `no`.

An example of authentication-methods configuration is shown below:

```
<authentication-methods>
  <auth-hostbased>
    <local-hostname name="host.example.com" />
  </auth-hostbased>
  <auth-gssapi allow-ticket-forwarding="yes"/>
  <auth-publickey>
    <key-selection policy="interactive-shy">
      <public-key type="plain" />
    </key-selection>
  </auth-publickey>
  <auth-keyboard-interactive />
  <auth-password>
    <password file="/path/filename" />
  </auth-password>
</authentication-methods>
```

hostbased-default-domain

This element specifies the host's default domain name (as `name`). This element is used to make sure the fully qualified domain name (FQDN) of the client host is transmitted to the server when using host-based user authentication.

The default domain name is appended to the short host name before transmitting it to the server. This is needed because some platforms (Solaris for instance) use the short format of the host name, and with that the signature cannot be created.

The allowed formats of the default domain names are: `.example.com` and `example.com` (without the leading dot). For example:

```
<hostbased-default-domain name=".example.com" />
```

compression

This element specifies whether the client sends the data compressed (PUT operation). When activated, compression is applied on-the-fly to all data sent out through the connection and on all channels in it.

The name of the compression algorithm and the compression level can be given as attributes. The `name` attribute can be defined as `none` (compression not used) or `zlib`, currently the only supported algorithm. By default, compression is not used.

For zlib compression, the `level` attribute can be given an integer from `0` to `9`. The default compression level is `6`, when compression is activated but no level is given (or level is set to `0`).

Example: to activate maximum level compression of sent data, make the following setting:

```
<compression name="zlib" level="9" />
```

Compression can also be activated per connection with command line tools. For information, see the sshg3(1), sftpg3(1) and scpg3(1) man pages.

Note that this `compression` setting does not affect received data (GET operations), but their compression is defined on the Secure Shell server. Tectia Server always uses compression level 6.

proxy

This element defines rules for HTTP proxy or SOCKS servers the client will use for connections. It has a single attribute: `ruleset`.

The format of the attribute value is a sequence of rules delimited by semicolons (`;`). Each rule has a format that resembles the URL format. In a rule, the connection type is given first. The type can be `direct`, `socks`, `socks4`, `socks5`, or `http-connect` (`socks` is a synonym for `socks4`). This is followed by the server address and port. If the port is not given, the default ports are used: 1080 for SOCKS and 80 for HTTP.

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` element is shown below. It causes the server to access the loopback address and the `ssh.com` domain directly, access `*.example` with HTTP CONNECT, and all other destinations with SOCKS4.

```
<proxy ruleset="direct:///127.0.0.0/8,*.ssh.com;
                http-connect://http-proxy.ssh.com:8080/*.example;
                socks://fw.ssh.com:1080/" />
```

idle-timeout

This element specifies how long idle time (after all connection channels are closed) is allowed for a connection before automatically closing the connection. The `time` is given in seconds. The `type` is always `connection`.

The default setting is 5 seconds. Setting a longer time allows the connection to the server to remain open even after a session (for example, **sshg3**) is closed. During this time, a new session to the server can be initiated without re-authentication. Setting the time to 0 (zero) terminates the connection immediately when the last channel to the server is closed.

```
<idle-timeout time="5" />
```

tcp-connect-timeout

This element specifies a timeout for the TCP connection. When this setting is made, connection attempts to a Secure Shell server are stopped after the defined time if the remote host is down or unreachable. This timeout overrides the default system TCP timeout, and this timeout setting can be overridden by defining a `tcp-connect-timeout` setting per connection profile (in the `profiles` settings) or per connection (on command line).

The `time` is given in seconds. The factory default is 5 seconds. Value 0 (zero) disables this feature and the default system TCP timeout will be used.

```
<tcp-connect-timeout time="5" />
```

keepalive-interval

This element specifies an interval for sending keepalive messages to the Secure Shell server. The `time` value is given in seconds. The default setting is 0, meaning that the keepalive messages are disabled.

```
<keepalive-interval time="0" />
```

exclusive-connection

The `exclusive-connection` element can be used to specify that a new connection is opened for each new channel. This setting takes one attribute `enable`, with value `yes` or `no`. The default is `no`, meaning that open connections are reused for new channels requested by a client.

server-banners

This element defines whether the server banner message file (if it exists) is visible to the user before login. The word `yes` or `no` is given as the value of the `visible` attribute. The default is `yes`.

To eliminate server banners:

```
<server-banners visible="no" />
```

forwards

This element contains `forward` elements that define whether X11 or agent forwarding (tunneling) are allowed on the client side.

forward

This element defines X11 or agent forwarding settings.

The `type` attribute defines the forwarding type (either `x11` or `agent`). The `state` attribute sets the forwarding `on`, `off`, or `denied`. If the forwarding is set as `denied`, the user cannot enable it on the command-line.

An example forward configuration, which denies X11 forwarding and allows agent forwarding globally, is shown below:

```
<forwards>
  <forward type="x11" state="denied" />
  <forward type="agent" state="on" />
</forwards>
```

For more information on using X11 and agent forwarding, see Section 6.3 and Section 6.4.

extended

This element is reserved for future use.

remote-environment

This element contains `environment` elements which define the environment variables to be passed to the server from the client side. The environment variables are then set on the server when requesting a command, shell or subsystem.

Note that the server can restrict the setting of environment variables.

environment

This element defines the name and value of the environment variables, and whether the Connection Broker should process the value. Possible attributes are `name`, `value`, and `format`.

An example remote environment configuration:

```
<remote-environment>
  <environment name="FOO" value="bar" />
  <environment name="QUX" value="%Ubaz" format="yes" />
  <environment name="ZAPPA" value="%Ubaz" />
</remote-environment>
```

You can use `%U` in the `value` to indicate a user name. When `format="yes"` is also defined, the Connection Broker processes the `%U` into the actual user name before sending it to the server.

Let's assume the user name is `joedoe` in this example. The example configuration results in the following environment variables on the server side, provided that the server allows setting the environment variables:

```
FOO=bar
QUX=joedoebaz
ZAPPA=%Ubaz
```

You can override the remote environment settings made in the configuration file if you use the **sshg3** command with the following arguments on the command-line client: `--remote-environment` or `--remote-environment-format`

For information on the command-line options, see sshg3(1).

server-authentication-methods

This `server-authentication-methods` element can be used to force the Connection Broker to use only certain methods in server authentication. This element can contain `auth-server-publickey` and `auth-server-certificate` elements (one of each). The order of these elements is free.

If only `auth-server-certificate` is specified, server certificate is needed. If no server certificate is received, connection fails.

If only `auth-server-publickey` is specified, (plain) server public key is needed. If no server public key is received, connection fails.

If both `auth-server-certificate` and `auth-server-publickey` are specified, server certificate is used if present. Otherwise server public key is used.

auth-server-certificate

The `auth-server-certificate` element specifies that certificates are used for server authentication.

auth-server-publickey

The `auth-server-publickey` element specifies that public host keys are used for server authentication.

Note

The host key policy settings have changed in version 6.1.4 and are now defined in the `auth-server-publickey` element.

The element takes attribute `policy` that defines how unknown server host keys are handled. It can have the following values:

- `strict`: Connect to the server only if the host key is found from the host key store and matches.

  If the policy is set to `strict`, the Connection Broker never adds host keys to the user's `.ssh2/hostkeys` directory upon connection, and refuses to connect to hosts whose key has changed. This provides maximum protection against man-in-the-middle attacks. However, it also means you must always obtain host keys via out-of-band means, which can be troublesome if you frequently connect to new hosts.

- `ask` (default): If the server host key is not found from the host key store, the user will be asked if he wants to accept the host key. If the host key has changed, the user is warned about it and asked how to proceed. If the client application is not able to ask the user (for example, **sftpg3** in batch mode, `-B`), the connection will be disconnected.

- `trust-on-first-use` or `tofu`: If the server host key is not found, it is stored to the user's `.ssh2/hostkeys` directory. If the host key has changed, the connection will be disconnected.

- `advisory`: Use of this setting effectively disables server authentication, which makes the connection vulnerable to active attackers.

  If the server host key is not found in the host key store, it will be added to the user's `.ssh2/hostkeys` directory without user interaction. If the host key has changed, the connection will be continued without user interaction. The incident will be audited if logging is enabled.

  When the policy is set to `advisory`, the keys from new hosts are automatically accepted and stored to the host key database without prompting acceptance from the user. However, changed host keys (from hosts whose keys are already in the database) are not stored, but they are accepted for that connection only.

  This setting should be used only if logging is enabled for the Connection Broker.

  ### Caution

  Consider carefully before setting the policy to `advisory`. Disabling the host-key checks makes you vulnerable to man-in-the-middle attacks.

In policy modes other than `strict`, if logging is enabled for the Connection Broker, Tectia Client will log information about changed and new host public keys with their fingerprints in the syslog (on Unix) or Event Viewer (on Windows).

Note

When FTP-SFTP conversion is used, accepting the host key cannot be prompted from the user. Either the policy must be set to `tofu` or the host keys of the Secure Shell tunneling and SFTP servers must be obtained beforehand and stored based on the IP addresses of the servers.

If the `policy` attribute is not defined, the host key policy is interpreted based on the values of the old `strict-host-key-checking`, `host-key-always-ask`, and `accept-unknown-host-keys` options as shown in Table A.2 below.

Note

In version 6.1.4 and later, the host key policy setting in the user-specific configuration file always takes precedence over the setting in the global configuration file.

Table A.2. Interpretation of old host key policy (Tectia Client 5.0.0-6.1.3) to new host key policy (Tectia Client 6.1.4 and later)

strict-host-key-checking	accept-unknown-host-keys	host-key-always-ask	Policy
-	-	-	**ask (default)**
enabled	-	-	**strict**
enabled	enabled	-	**strict**

strict-host-key-checking	accept-unknown-host-keys	host-key-always-ask	Policy
enabled	enabled	enabled	**ask**
enabled	-	enabled	**ask**
-	enabled	-	**trust on first use**
-	enabled	enabled	**ask**
-	-	enabled	**ask**

authentication-method

The `server-authentication-methods/authentication-method` element specifies an authentication method `name`. This element is included for backwards compatibility. Use the `auth-server-*` elements instead.

```
<server-authentication-methods>
  <authentication-method name="publickey" />
  <authentication-method name="certificate" />
</server-authentication-methods>
```

An example `server-authentication-methods` element is shown below:

```
<server-authentication-methods>
  <auth-server-publickey policy="ask" />
  <auth-server-certificate />
</server-authentication-methods>
```

authentication-success-message

This setting defines whether the `AuthenticationSuccessMsg` message is shown. The `authentication-success-message` element takes the attributes `enable` and `delay`.

`Enable` can be either `yes` or `no`. The default is `yes`, meaning that the messages are output and logged.

`Delay` takes a numerical value, which corresponds to how many seconds the authentication success message is shown. The default value is `2`. When set to `0`,the message is only logged, not shown.

disconnect-message

Message that displays as you disconnect. The value of the disconnect message is a string. The message may contain any number of the following substitution variables:

- `time`: Time of disconnect.
- `random`: 16 random hexadecimal digits.
- `random4`: 4 random hexadecimal digits.
- `random8`: 8 random hexadecimal digits.
- `random16`: Same as *`random`*.
- `pid`: Process id of sshg3.

- `broker_pid`: Broker process id.
- `conn_id`: Connection id.
- `session_id`: Session id.
- `target_host`: Target server name.
- `target_port`: Target server port.

If any of the random variables are used in a disconnect message, the random values are displayed to the user before authentication successful message. If the values before the connection and after the disconnect differ, somebody may be spoofing your connection.

Disconnect message is turned off by default.

keyboard-interactive

This setting contains the attribute `prefix`, which takes a string value. The prefix value will show before of the keyboard-interactive prompt when set. The prefix attribute's value can take the following substitution variables:

- `time`
- `host`
- `port`
- `user`
- `connid`

The default value for the prefix is `${host}>`.

sftpg3-mode

This setting defines how the **sftpg3** client behaves when transferring files. The `sftpg3-mode` element takes attribute `compatibility-mode` with the following values:

- `tectia` (the default) - **sftpg3** fransfers files recursively, meaning that files from the current directory and all its subdirectories are transferred.
- `ftp` - the **get/put** commands are executed as **sget/sput** meaning that they transfer a single file; and commands **mget/mput** have recursion depth set to 1 meaning that they only transfer files from the specified directory, not from subdirectories.
- `openssh` - commands **get/put/mget/mput** behave alike, and the recursion depth is set to 1, meaning that only files from the specified directory are transferred, not from subdirectories.

The recursion depth can be overridden by using the **sftpg3** client's commands **get/put/mget/mput** with command-line option `--max-depth="LEVEL"`. For more information, see sftpg3(1).

terminal-selection

This element defines how the Tectia terminal behaves when the user selects text with double-clicks. The element takes one attribute: `selection-type`, whose value can be:

`select-words` - double-clicking selects one word at a time, space and all punctuation characters are used as delimiters. This is the default.

`select-paths` - selects strings of characters between spaces, meaning a selection is extended over characters `\/.-_`, so that for example a path to a file can be selected by double-clicking anywhere in the path.

terminal-bell

This element defines whether Tectia terminal repeats audible notifications from the destination server. This option is only applied to connections with Unix servers. The element takes one attribute, `bell-style`, whose value can be:

`none` - no audible notifications are used

`pc-speaker` - the user's PC speakers beep when an audible notification is indicated by the destination server

`system-default` - the Tectia terminal sounds the default alerts defined in the system on the destination server. This is the default.

close-window-on-disconnect

This element defines that also the Tectia terminal window is to be closed while disconnecting from a server session by pressing **CTRL+D**. The element takes one attribute, `enable`, whose value can be `yes` or `no`. The default is `no` meaning that **CTRL+D** closes only the server connection but the Tectia terminal window remains open.

quiet-mode

This setting defines whether the command line clients should suppress warnings, error messages and authentication success messages. The `quiet-mode` element takes attribute `enable` with value `yes` or `no`. The default is `no`, meaning that the errors and messages are output and logged.

The `quiet-mode` element affects command line tools **scpg3**, **sshg3**, and **sftpg3**. Enabling the quiet mode here with setting `quiet-mode enable="yes"` is the same as running these clients with option `-q`. Note that the `-q` command line parameter will take priority over the `quiet-mode` element set in this configuration file.

checksum

The `checksum` element can be used to define a default setting for comparing checksums or checkpoint to determine the point in the existing destination file where the file transfer can be resumed. With this option SHA-1 or SHA-2 checksums can be used instead of MD5 or checksum forced also on small files. By default checksums are not checked for files smaller than default buffer size 32kB.

The `checksum` element takes attribute `type`, whose value can be:

`yes|YES` - checksums are checked on files larger than 32kB. This is the default value (uses SHA1 checksums in FIPS mode, MD5 checksums otherwise).

`no|NO` - checksums are not used.

`md5|MD5` - MD5 checksums are checked on files larger than 32kB. When the `--fips` parameter is set with the command line clients **scpg3** and **sftpg3**, this hash is not used instead "Warning: MD5 is not FIPS certified -> switching to SHA1." is logged.

`sha1|SHA1` - SHA1 checksums are checked on files larger than 32kB. When the `--fips` parameter is set with the command line clients **scpg3** and **sftpg3**, this hash is used.

`sha256|SHA256` - SHA256 checksums are checked on files larger than buffer size.

`sha512|SHA512` - SHA512 checksums are checked on files larger than buffer size.

`md5-force|MD5-FORCE` - MD5 checksums are forced on all files, except when the `--fips` parameter is set with the command line tools **scpg3** and **sftpg3**.

`sha1-force|SHA1-FORCE` - SHA1 checksums are forced on all files.

`sha256-force|SHA256-FORCE` - SHA256 checksums are forced on all files.

`sha512-force|SHA512-FORCE` - SHA512 checksums are forced on all files.

`checkpoint|CHECKPOINT` - checkpointing is forced on large files that are transferred one by one.

Note

If the Connection Broker is started in FIPS mode and the `md5` attribute is defined in the configuration file, but **scpg3** or **sftpg3** are not started with the `--fips` parameter or system-wide FIPSMODE is not set, then `md5` is used.

Note that checksums can also be defined with the command line clients **scpg3** and **sftpg3**, or with environment variables. The order of priority of the three checksum settings (in case they are different) is as follows, the later one always overwrites the previous value:

- `checksum` setting in the configuration file
- `SSH_SFTP_CHECKSUM_MODE` environment variable
- Command line arguments

address-family

The `address-family` element defines the IP address family. Give the address family as the value of the `type` attribute. Tectia Client will operate using IPv4 (`inet`) addressing, IPv6 (`inet6`), or both (`any`). The default value for `type` is `any`.

The `profiles` Element

The `profiles` element defines the connection profiles for connecting to the specified servers. Element `profiles` can contain multiple `profile` elements. Typically, each profile defines the connection rules

to one server. A generic profile, specifying for example required algorithms, can be used with the **sshg3**, **scpg3** or **sftpg3** command-line option `--template-profile` when connecting to servers. The settings in the `profile` element override the default connection settings.

When a profile is used for the connection, the settings in the profile override the default settings. See the section called "The `default-settings` Element".

profile

The `profile` element defines a connection profile. It has the following attributes: `id`, `name`, `host`, `port`, `protocol`, `host-type`, `connect-on-startup`, `user`, and `gateway-profile`.

The profile `id` must be a unique identifier that does not change during the lifetime of the profile.

An additional `name` can be given to the profile. This is a free-form text string. The name can be used for connecting with the profile on the command line, so define a unique name for each profile.

The `host` attribute defines the address of the Secure Shell server host and it is a mandatory setting. The address can be either an IP address or a domain name. The value `host="*"` can be used to prompt the user to enter the host address when starting the session.

The `port` is a mandatory setting. It defines the port number of the Secure Shell server listener. The default port is `22`.

The `protocol` is a mandatory setting. It defines the used communications protocol. Currently the only allowed value is `secsh2`.

If you want to make the connection specified by the profile automatically when the Connection Broker is started, set the value of the `connect-on-startup` attribute to `yes`. In this case, give also the `user` attribute (the user name the connection is made with). You also need to set up some form of non-interactive authentication for the connection.

The `host-type` attribute sets the server type for ASCII (text) file transfer. This specifies the line break convention that is used for ASCII files. The default value is `default`, meaning that the line break convention is determined by the local platform. If the client is running on Windows, Windows compatible line breaks (CR + LF, `'\r\n'`) are used. If the client is running on any other platform, Unix compatible line breaks (LF, `'\n'`) are used. Other possible values for `host-type` are `windows` (for Windows remote host) and `unix` (for Unix remote host). Define the value if you are using any other server than Tectia Server.

The `user` attribute specifies the user name for opening the connection. The value `"%USERNAME%"` can be used to apply the user name of the currently logged in user. The value `user="*"` can be used to prompt the user to enter the user name when logging in. When the `user` attribute is not defined, the user name defined in the default connection settings will be used.

The `gateway-profile` attribute can be used to create nested tunnels. The tunnels defined under the `local-tunnel` element of the profile, and the tunnels defined under `filter-engine` and `static-tunnels` that refer to the profile can be nested. The profile name through which the connection is made is given as the value of the attribute. The first tunnel is created using the gateway host profile and from there the second tunnel is created to the host defined in this profile.

hostkey

This element gives the path to the remote server host public key file as a value of the `file` attribute.

Alternatively, the public key can be included as a base64-encoded ASCII block.

ciphers

This element defines the ciphers used with this profile. See **ciphers** for details.

macs

This element defines the MACs used with this profile. See **macs** for details.

kexs

This element defines the KEXs used with this profile. See **kexs** for details.

hostkey-algorithms

This element defines the hostkey signature algorithms used with this profile. See **hostkey-algorithms** for details.

rekey

This element defines the rekeying settings used with this profile. See **rekey** for details.

authentication-methods

This element defines the authentication methods used with this profile. See **authentication-methods** for details.

user-identities

This element specifies the identities used in user public-key authentication. In contrast to the `key-stores` element that specifies all the keys that are available for the Connection Broker, this element can be used to control the keys that are attempted in authentication when this connection profile is used and to specify the order in which they are attempted.

The `user-identities` element can contain multiple `identity` elements. When multiple `identity` elements are used, they are tried out in the order they are listed.

identity

The `identity` element has the following attributes: `file`, `hash`, `identity-file`, `id`, and `data`.

The `file` attribute specifies the path to the public-key file (primarily) or to a certificate. Enter the full path and file name as the value.

The `hash` attribute is used to enter the hash of the public key that will be used to identify the related private key. The key must be available for the Connection Broker The public

key hashes of the available keys can be listed with the **ssh-broker-ctl** tool. See also ssh-broker-ctl(1).

The `identity-file` attribute is reserved for future use.

The `id` attribute is reserved for future use.

The `data` attribute is reserved for future use.

An example `user-identities` element is shown below:

```
<user-identities>
  <identity file="$HOME/user/.ssh2/id_rsa_3072_a" />
  <identity file="C:\%username-without-domain%\private_keys\id_rsa_4096_a" />
  <identity hash="#a8edd3845005931aaa658b5573609e7d31e23afd" />
</user-identities>
```

compression

This element defines the compression settings used with this profile. See **compression** for details.

proxy

This element defines the HTTP proxy and SOCKS server settings used with this profile. See **proxy** for details.

If `gateway-profile` has been defined for this profile, the proxy setting is ignored and the default proxy setting or the proxy setting of the gateway profile is used instead.

idle-timeout

This element defines the idle timeout settings used with this profile. See **idle-timeout** for details.

tcp-connect-timeout

This element defines the TCP connection timeout for this profile. The timeout is used to terminate connection attempts to Secure Shell servers that are down or unreachable. The default value is 5 seconds. See **tcp-connect-timeout** for details.

keepalive-interval

This element defines an interval for sending keepalive messages to the Secure Shell server. The setting applies to this profile. The default value is 0, meaning that no keepalive messages are sent. See **keepalive-interval** for details.

exclusive-connection

This element defines whether a new connection is opened for each new channel when a connection is made with this profile. This setting takes one attribute `enable`, with value `yes` or `no`. The default is `no`, meaning that open connections are reused for new channels requested by a client. See also **exclusive-connection** .

server-banners

This element defines the server banner setting used with this profile. See **server-banners** for details.

forwards

This element defines the forwards allowed with this profile. See **forwards** for details.

tunnels

The `tunnels` element defines the tunnels that are opened when a connection with this profile is made. The element can contain multiple `local-tunnel` and `remote-tunnel` elements.

local-tunnel

This element defines a local tunnel (port forwarding) that is opened automatically when a connection is made with the connection profile. It has five attributes: `type`, `listen-port`, `listen-address`, `dst-host`, `dst-port`, and `allow-relay`.

The `type` attribute defines the type of the tunnel. This can be `tcp` (default, no special processing), `ftp` (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session), or `socks` (Tectia Client will act as a SOCKS server for other applications, creating forwards as requested by the SOCKS transaction).

The `listen-port` attribute defines the listener port number on the local client.

The `listen-address` attribute can be used to define which network interfaces on the client should be listened. Its value can be an IP address belonging to an interface on the local host. Value `0.0.0.0` listens to all interfaces. The default is `127.0.0.1` (localhost loopback address on the client). Setting any other value requires setting `allow-relay="yes"`.

For `address-family` option `inet6`, the default listen address is `::1`. To listen on all interfaces, specify `::`. For `address-family` option `any`, the listen address is both `127.0.0.1` and `::1` by default; to listen on all interfaces, specify `::`.

Whenever a connection is made to the specified listener, the connection is tunneled over Secure Shell to the remote server and another connection is made from the server to a specified destination host and port (`dst-host`, `dst-port`). The connection from the server onwards will not be secure, it is a normal TCP connection.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = server host).

The `allow-relay` attribute defines whether connections to the listened port are allowed from outside the client host. The default is `no`. If you use `allow-relay="yes"`, it will check also the `listen-address` setting.

For more information on using local tunnels, see Section 6.1.

remote-tunnel

This element defines a remote tunnel (port forwarding) that is opened automatically when a connection is made with the connection profile. It has four attributes: `type`, `listen-port`, `listen-address`, `dst-host`, `dst-port`, and `allow-relay`.

The `type` attribute defines the type of the tunnel. This can be either `tcp` (default, no special processing) or `ftp` (temporary forwarding is created for FTP data channels, effectively securing the FTP session between the client and server).

The `listen-port` attribute defines the listener port number on the remote server.

The `listen-address` attribute can be used to define which network interfaces on the server should be listened. Its value can be an IP address belonging to an interface on the server host. Value `0.0.0.0` listens to all interfaces. The default is `127.0.0.1` (localhost loopback address on the server). Setting any other value requires that `allow-relay="yes"`.

For `address-family` option `inet6`, the default listen address is `::1`. To listen on all interfaces, specify `::`. For `address-family` option `any`, the listen address is both `127.0.0.1` and `::1` by default; to listen on all interfaces, specify `::`.

Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified destination host and port (`dst-host`, `dst-port`). The connection from the client onwards will not be secure, it is a normal TCP connection.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = client host).

The `allow-relay` attribute defines whether connections to the listener port are allowed from outside the server host. The default is `no`.

For more information on using remote tunnels, see Section 6.2.

extended

This element is reserved for future use.

remote-environment

This element defines the remote environment settings used with this profile. Within the `remote-environment` element, define an `environment` element for each environment variable to be passed to the server. See **remote-environment** for details.

server-authentication-methods

This element defines the server authentication methods allowed with this profile. See **server-authentication-methods** for details.

password

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

The password can be given directly in the `string` attribute, or a path to a file containing the password can be given in the `file` attribute, or a path to a program or a script that outputs the password can be given in the `command` attribute.

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

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

Caution

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

Note

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

An example connection profile is shown below:

```
<profile name="rock"
         id="id1"
         host="rock.example.com"
         port="22"
         connect-on-startup="no"
         user="doct">

  <hostkey file="key_22_rock.pub">
  </hostkey>

  <authentication-methods>
    <auth-publickey />
    <auth-password />
  </authentication-methods>

  <server-authentication-methods>
    <auth-server-publickey policy="strict" />
  </server-authentication-methods>

  <server-banners visible="yes" />

  <forwards>
    <forward type="agent" state="on" />
    <forward type="x11" state="on" />
```

```
  </forwards>

  <tunnels>
    <local-tunnel type="tcp"
                  listen-port="143"
                  dst-host="imap.example.com"
                  dst-port="143"
                  allow-relay="no" />
  </tunnels>

  <remote-environment>
    <environment name="FOO" value="bar" />
    <environment name="QUX" value="%Ubaz" format="yes" />
    <environment name="ZAPPA" value="%Ubaz" />
  </remote-environment>

</profile>
```

The `static-tunnels` Element

The `static-tunnels` setting is used to configure the behavior of the automatic tunnels. You can create listeners for local tunnels automatically when the Connection Broker starts up. The actual tunnel is formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.

The `static-tunnels` element can contain any number of `tunnel` elements.

tunnel

The `tunnel` element specifies a static tunnel. It has the following attributes: `type`, `listen-port`, `listen-address`, `dst-host`, `dst-port`, `allow-relay`, and `profile`.

The `type` attribute defines the type of the tunnel. This can be either `tcp` or `ftp`.

- `tcp` specifies a listener for generic TCP tunneling
- `ftp` specifies a listener for FTP tunneling (also the FTP data channels are tunneled)

The `listen-port` attribute defines the listener port number on the local client.

The `listen-address` attribute can be used to define which network interfaces on the client should be listened. Its value can be an IP address belonging to an interface on the local host. Value `0.0.0.0` listens to all interfaces. The default is `127.0.0.1` (localhost loopback address on the client). Setting any other value requires that `allow-relay="yes"`.

For `address-family` option `inet6`, the default listen address is `::1`. To listen on all interfaces, specify `::`. For `address-family` option `any`, the listen address is both `127.0.0.1` and `::1` by default; to listen on all interfaces, specify `::`.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = server host).

The `allow-relay` attribute defines whether connections to the listened port are allowed from outside the client host. The default is `no`.

The `profile` attribute specifies the connection profile ID that is used for the tunnel.

```
<static-tunnels>
  <tunnel type="tcp"
          listen-address="127.0.0.1"
          listen-port="9000"
          dst-host="st.example.com"
          dst-port="9000"
          allow-relay="no"
          profile="id1" />
</static-tunnels>
```

The `gui` Element

The `gui` element is used to adjust the Tectia terminal GUI settings. The `gui` element takes the following attributes: `hide-tray-icon`, `show-exit-button`, and `show-admin`. All of these must have `yes` or `no` as the value.

The `hide-tray-icon` attribute controls whether the Tectia icon is displayed in the notification area of the Windows taskbar (also known as the system tray). The default is `no` (the tray icon is displayed).

The `show-exit-button` attribute controls whether the **Exit** command is displayed in the shortcut menu of the Tectia icon. The default is `yes`.

The `show-admin` attribute defines whether the **Configuration** command is displayed in the Tectia icon shortcut menu. The default is `yes`. If the button is not displayed, the Tectia Connections Configuration GUI can be started by running `ssh-tectia-configuration.exe`, located by default in directory `"<INSTALLDIR>\SSH Tectia Broker"`.

```
<gui hide-tray-icon="no"
     show-exit-button="yes"
     show-admin="yes"
```

The `filter-engine` Element

The `filter-engine` element defines the filter rules for transparent TCP tunneling.

Note

The `filter-engine` element is read from the global configuration file, if such a file is available. Only when the global configuration file does not contain the `filter-engine` element, this element is read from the user-specific configuration file.

On Unix, the global configuration is stored as `/etc/ssh2/ssh-broker-config.xml`, and on Windows as `"<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml"`, where `<INSTALLDIR>` indicates the default Tectia installation directory on Windows (see Section 1.1.2).

For configuration examples, see these sample files:

- On Unix: `etc/ssh2/ssh-broker-config-example-capture.xml` and `etc/ssh2/ssh-broker-config-example.xml`
- On Windows: "`<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config-example.xml`"

The top level element is `filter-engine`. It has the following attributes: `capture-enabled`, `ip-generate-start`, `ip6-generate-start` and `ftp-filter-at-signs` (currently unused).

`capture-enabled` attribute defines whether transparent TCP tunneling is active and capturing application connections for tunneling. The default is `yes`.

The `ip-generate-start` attribute defines the start address of the pseudo IPv4 address space. Pseudo IPs are generated by the Connection Broker when applications do the DNS query through the SSH connection capture component.

The `ip6-generate-start` attribute is similar to `ip-generate-start`, but it defines the start address of the pseudo IPv6 address space.

Note

Under the `filter-engine` element there can be any amount of `rule` elements. The order of the elements is important, because the filter engine uses the elements in the order they were specified in the configuration file.

rule

The `rule` element specifies how a filtered connection will be handled. It has the following attributes: `application`, `host`, `ip-address`, `pseudo-ip`, `ports`, `action`, `profile-id`, `destination`, `destination-port`, `hostname-from-app`, `username-from-app`, `fallback-to-plain`.

The `application` attribute can be used to specify one or more applications to which the rule is applied. This can be a regular expression using the egrep syntax. For information on the syntax, see Appendix C.

The `host` attribute specifies a target host name. It can be a regular expression using the egrep syntax.

The `ip-address` attribute specifies the target host IP address. It can be a regular expression using the egrep syntax. In this case the Connection Broker does the string matching with the assumption that the IP address is written in its canonical form. If both the host name and the IP address are defined, the `host` attribute takes precedence and the `ip-address` attribute is ignored.

The `pseudo-ip` setting has the following effects when the `ip-address` is left empty and the `host` matches:

- When `pseudo-ip="yes"`, the Connection Broker assigns a pseudo IP address for the target host and Tectia Server resolves the real IP address. The pseudo IP addresses should be used when accessing an internal network from the outside, because name resolution for the machines in the internal network is not available from the outside.

- When `pseudo-ip="no"`, a normal DNS query is made for the target host name. The default value is `no`.

`pseudo-ip` is always used on Windows regardless of this setting's value.

The `ports` attribute can be a single port or a range. A range is specified with a hyphen between two integers (for example `"21-25"`).

Note

For FTP-SFTP conversion, always specify the port unambiguously if fallback mode is set. Do not use an asterisk (*), because it causes problems in passive mode file transfer when connected to a plaintext FTP server.

The `action` attribute specifies the action to be done when a filter matches. Its value can be `DIRECT`, `BLOCK`, `TUNNEL`, `FTP-TUNNEL`, or `FTP-PROXY`.

- `DIRECT` causes the connection to be made directly as plaintext without tunneling or FTP-SFTP conversion.
- `BLOCK` causes the connection to be blocked.
- `FTP-TUNNEL` activates transparent FTP tunneling
- `TUNNEL` activates transparent TCP tunneling
- `FTP-PROXY` causes the FTP-SFTP conversion to start and a connection to be made to the Secure Shell SFTP server.

The `profile-id` attribute can be used to specify the connection profile that defines the connection settings.

If the `profile-id` attribute is left empty and `hostname-from-app="yes"` is specified, the Secure Shell connection is made to the server specified by the client application using default settings. If a `profile-id` is specified and also `hostname-from-app="yes"` is specified, or the referred profile has `*` (an asterisk) or empty as the value of the `host` attribute, the Secure Shell connection is made to the server specified by the client application using the profile settings.

The `destination` and `destination-port` attributes can be used to define a static destination address and port number that will be used as the end point of the connection instead of the original address and port given by the application.

The `hostname-from-app` attribute defines whether the Connection Broker should extract the Secure Shell server's host name from data sent by the application, or use a Secure Shell server defined by the connection profile in `profile-id`. The value is `yes` or `no`, and the default is `no`.

When `hostname-from-app="no"`, the tunnel will be created to the Secure Shell server specified in the connection profile referred in the `profile-id` attribute. Note that with transparent tunneling, the connection from the Secure Shell server to the final destination application will be unsecured and

in plaintext. To achieve end-to-end security, the Secure Shell server should reside on the same host as the application.

When `hostname-from-app="yes"`, the tunnel will be created to the destination server specified by the application. This setting can be used with both FTP and TCP tunneling and FTP-SFTP conversion. When using `hostname-from-app="yes"`, it is no longer necessary to create a separate connection profile for each destination host. Note that this requires that a Secure Shell server is installed to each destination server (or that `fallback-to-plain` is enabled to allow direct connections to those servers that do not have Secure Shell installed).

The `username-from-app` attribute defines whether the FTP tunneling or FTP-SFTP conversion extracts the user name from data sent by the FTP application. The value is `yes` or `no`. The default is `no`.

When `username-from-app="yes"`, the user name received from the FTP client application is used. This setting can be used with FTP tunneling and FTP-SFTP conversion. This setting will override any user name settings made in a related connection profile. When `username-from-app="no"`, the user name is taken from the connection profile defined with the `profile-id` attribute.

The `fallback-to-plain` attribute can be used to define whether a direct (unsecured) connection is used if creating the tunnel fails or the connection to the Secure Shell server fails. The default value is `no`. Normally, when the secured connection fails when applying a filter rule, the Connection Broker will return an error about not being able to establish a connection. In FTP-SFTP conversion on Unix, `fallback-to-plain` requires that option `-F` is used with the `ssh-capture` command.

Note

Do not enable the `fallback-to-plain` and `pseudo-ip` options at the same time. If they both are enabled, and a secure connection fails, the application will try a direct connection with the pseudo IP, which will not work.

The `windows-capture` Element

The supported `windows-capture` options and their default values:

```
<windows-capture
      trusted-user = "nt authority\system"
      hostname-filtering= "always"
      trusted-broker-start-on-boot = "yes"
      trusted-broker-enabled = "yes"
      trusted-broker-offline-action = "direct"
      user-broker-enabled = "yes"
      user-broker-offline-action = "direct"/>
```

trusted-user

Defines the user whose broker is treated as the trusted broker. The trusted broker has priority over user brokers in decicions about tunneling/blocking all the connections on the system.

If you set this to any non service account that has a password, and `trusted-broker-start-on-boot` is `"yes"`, you must run `ssh-broker-ctl capture-driver reconfig` as that user. That user

must also be in the Administrators group. Otherwise the trusted broker automatic launch task will not be updated. Updating any other windows-capture setting will succeed regardless of the user you are running as, as long as you run it as the Administrator.

If you want auto start with a non-Administrator trusted user, then you must manually edit the Task Scheduler task TectiaTrustedBroker to have the correct user. After that run `ssh-broker-ctl capture-driver reconfig --force` (as any user with Administrator privileges). The trusted broker will be launched automatically only starting from next boot. If you want to run it immediately, set the task to `Run` in the Task Scheduler.

`hostname-filtering`

Possible values: `never`, `trusted-broker`, `always`

If set to never, hostname filtering will be entirely disabled and filtering rules with a hostname will never be matched. If set to trusted-broker, only the trusted broker can do hostname filtering. With always, all users are allowed to do hostname filtering (given that the trusted broker is enabled and running).

`trusted-broker-start-on-boot`

If enabled, the trusted broker is launched automatically by the Windows Task Scheduler as the trusted-user. This broker always has a special socket address **capture-trusted-broker**.

Administrators must have the permission `Log on as a batch job`. Otherwise the trusted broker cannot be started automatically. `Log on as a batch job` is enabled by default.

`trusted-broker-enabled`

Enables the usage of the trusted broker feature. If disabled, no broker will be treated as the trusted broker (see trusted-user).

If the trusted broker is enabled and set up, it will always have the first say on any connection before any user broker.

`trusted-broker-offline-action`

The default action for all connections in case the trusted broker enabled is enabled but not running. Possible values are `direct` and `block`.

`user-broker-enabled`

If enabled, allows any user to filter with their own broker and filtering rules. User brokers can only filter connections that are run by the same user. Hostname-based filtering is available to user brokers only if the trusted broker is enabled and running.

`user-broker-offline-action`

Same as `trusted-broker-enabled`, but for the user broker.

The `logging` Element

The `logging` element changes the logging settings that define the log event severities and logging facilities. The element contains one or more `log-target` and `log-events` elements.

log-target

This element specifies the log target for auditing. By default, the broker does not log anything. This element can be used to direct log data to a file or syslog.

The `log-target` element can have `file` and `type` as attributes.

The `type` attribute specifies the logging facility where the audit data is output to. The value can be `file`, `syslog` or `discard`.

The `file` attribute sets the file system path where the audit data is written to. If the `type` attribute has `syslog` or `discard` set, the `file` attribute is not allowed.

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.

The element can also contain one or more `log-target` elements. When defined here, the `log-target` element will override the definition given in the `logging/log-target`.

For the events, `facility` and `severity` can be set as attributes. The events itself should be listed inside the `log-events` element.

The facility can be `normal`, `daemon`, `user`, `auth`, `local0`, `local1`, `local2`, `local3`, `local4`, `local5`, `local6`, `local7`, or `discard`. Setting the facility to `discard` causes the server to ignore the specified log events.

On Windows, only the `normal` and `discard` facilities are used.

The severity can be `informational`, `notice`, `warning`, `error`, `critical`, `security-success`, or `security-failure`.

Any events that are not specifically defined in the configuration file will use the default values. The defaults can be overridden for all remaining events by giving an empty `log-events` element after all other definitions and by setting a severity value for it.

In the names of log events, the characters '*' and '?' can be used as wildcards.

For a complete list of log events, see Appendix D.

An example logging configuration that logs all events, which are programmed to be logged by default, both to `/tmp/foo` and to `syslog`.

```
<logging>
  <log-target file="/tmp/foo" />
  <log-target type="syslog" />
</logging>
```

An example logging configuration in which events are logged to `/tmp/foo`, except those whose event name matches "`Key_store_*`", which will be discarded.

```
<logging>
  <log-target file="/tmp/foo" />
  <log-events facility="discard">
    Key_store_*
  </log-events>
</logging>
```

A.3 Backup of Configuration Files

Before you start upgrades or creating test configurations, make sure you have a backup of the Connection Broker configuration files where you have made modifications.

The user-specific Connection Broker configuration file is by default located in `$HOME/.ssh2/ssh-broker-config.xml` on Unix and in `%APPDATA%\SSH\ssh-broker-config.xml` on Windows. Each time the Connection Broker configuration file is saved, a backup (`ssh-broker-config.xml.bak`) of the old configuration file is stored in the same directory. In case you need to return to using the backed up file, copy it back to the original location under the original name.

During Tectia upgrades on Windows, a backup copy is automatically made of the earlier Connection Broker configuration files and stored in the user-specific directory:

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

where

`<version>` is the Tectia release

`<date>` is the date of the upgrade.

A.4 Connection Broker Configuration File Quick Reference

This Appendix contains a quick reference to the elements of the Connection Broker configuration file, `ssh-broker-config.xml`. The quick reference is divided into four tables:

- Table A.3: The `general` element
- Table A.4: The `default-settings` element
- Table A.5: The `profiles` element
- Table A.6: Other elements (`static-tunnels`, `gui`, and `logging`)

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-broker-config(5).

The element hierarchy is expressed with slashes ('/') between parent and child elements.

Table A.3. `ssh-broker-config.xml` Quick Reference - the `general` element

Element	Attributes and their values	Description
crypto-lib	`mode = "`**`standard`**`|fips"`	Cryptographic library mode: standard or FIPS 140-2 certified.
cert-validation	`end-point-identity-check = "`**`yes`**`|no|ask"`	Client will verify server's host name or IP address against the server host certificate
	`default-domain =` *`domain_name`*	Default domain part of the remote system name
	`http-proxy-url =` *`HTTP_proxy`*	HTTP proxy for making queries for certificate validity
	`socks-server-url =` *`SOCKS_server`*	SOCKS server for making queries for certificate validity
	`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`* (default: "**`10`**")	Maximum length of certification paths when validating certificates
cert-validation / **ldap-server**	`address =` *`LDAP_server_address`*	LDAP server address for fetching CRLs and/or subordinate CA certificates
	`port =` *`port_number`* (default: "**`389`**")	LDAP server port for fetching CRLs and/or subordinate CA certificates
cert-validation / **ocsp-responder**	`url =` *`URL_address`*	OCSP (Online Certificate Status Protocol) responder service address
	`validity-period =` *`seconds`* (default: "**`0`**")	Time period during which new OCSP queries for the same certificate are not made (the old result is used)
cert-validation / **crl-prefetch**	`url =` *`LDAP_URL|HTTP_URL|file_URL`*	Tectia Client periodically downloads a CRL from this URL
	`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 certification authority (CA) used in server authentication
	`file =` *`path`*	Path to the X.509 CA certificate file
	`disable-crls = "yes|no"`	Disable CRL checking
	`use-expired-crls =` *`seconds`* (default: "**0**")	Time period for using expired CRLs
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
key-stores / **key-store**	`type = "mscapi|pkcs11|software|zos-saf"`	Key store type
	`init =` *`init_info`*	Key-store-provider-specific initialization info
key-stores / **user-keys**	`directory =` *`path`*	Directory where the user private keys are stored
	`passphrase-timeout =` *`seconds`* (default: "**0**")	Time after which the passphrase-protected private key will time out
	`passphrase-idle-timeout =` *`seconds`* (default: "**0**")	Time after which the passphrase-protected private key will time out unless the user accesses or uses the key
key-stores / **identification**	`file =` *`path`*	Location of the identification file that defines the user keys
	`base-path =` *`path`*	Directory where the identification file expects the user private keys to be stored
	`passphrase-timeout =` *`seconds`* (default: "**0**")	Time after which the user must enter the passphrase again
	`passphrase-idle-timeout =` *`seconds`* (default: "**0**")	Time after which the passphrase times out if there are no user actions
user-config-directory	`path =` *`path`* (default: "**%USER_CONFIG_DIRECTORY%**")	Non-default location of user-specific configuration files
file-access-control *(Unix only)*	`enable = "yes|no"`	Enable checking of file access permissions defined for global and user-specific configuration files and private keys files

Element	Attributes and their values	Description
protocol-parameters	`threads = number` (if set to `0`, default value is used)	The number of threads the protocol library uses (fast path dispatcher threads)
known-hosts	`path = path`	Non-default location of known hosts file or directory
	`file = path`	Location of OpenSSH-style `known_hosts` file
	`directory = path`	Non-default directory for storing known host keys
	`filename-format = "hash|plain|default"` (`"default" = "hash"`)	The format in which new host key files will be stored

Table A.4. `ssh-broker-config.xml` Quick Reference - the `default-settings` element

Element	Attributes and their values	Description
	`user = user_name`	Default user name to be used when connecting to remote servers
ciphers / **cipher**	`name = cipher_name`	A cipher that the client requests for data encryption
macs / **mac**	`name = MAC_name`	A MAC that the client requests for data integrity verification
kexs / **kex**	`name = KEX_name`	A KEX that the client requests for the key exchange method
hostkey-algorithms / **hostkey-algorithm**	`name = hostkey-algorithm_name`	A host key signature algorithm to be used in server authentication with host keys or certificates
rekey	`bytes = number` (default: `"1000000000"` (1 GB))	Number of transferred bytes after which key exchange is done again
authentication-methods / **auth-hostbased**	-	Host-based authentication will be used
authentication-methods / **auth-hostbased** / **local-hostname**	`name = host_name`	Local host name that is advertised to the remote server during host-based authentication
authentication-methods / **auth-password**	-	Password authentication will be used
authentication-methods /	-	Public-key authentication will be used

Element	Attributes and their values	Description
auth-publickey	`signature-algorithms = comma-separated_list`	Public-key signature algorithms used for client authentication
authentication-methods / **auth-publickey** / **key-selection**	`policy = "automatic|interactive-shy"`	Key selection policy used by the client when proposing user public keys to the server
authentication-methods / **auth-publickey** / **key-selection** / **public-key**	`type = "plain|certificate"` (by default, both are tried)	Only plain public keys or only certificates are tried during public-key authentication
authentication-methods / **auth-publickey** / **key-selection** / **issuer-name**	`name = certificate_issuer_name`	Client-side user certificates can be filtered by comparing this name to the certificate issuers requested or accepted by the server
	`match-server-certificate = "yes|no"`	The Connection Broker tries matching the user certificate issuer name to the server certificate issuer name
authentication-methods / **auth-gssapi**	-	GSSAPI will be used in authentication
	`dll-path = path` (*ignored on Windows*)	Location of the necessary GSSAPI libraries
	`allow-ticket-forwarding = "yes|no"`	Allow forwarding the Kerberos ticket over several connections
authentication-methods / **auth-keyboard-interactive**	-	Keyboard-interactive methods will be used in authentication
hostbased-default-domain	`name = domain_name`	Host's default domain name that is appended to the short host name before transmitting it to the server
compression	`name = "none|zlib"`	Compress the data that the client sends
	`level = [0 to 9]` (default: "**0**" (= level 6))	For `zlib`, compression level.
proxy	`ruleset = rule_sequence`	Rules for HTTP proxy or SOCKS servers the client will use for connections
idle-timeout	`type = "connection"`	Idle timeout is always defined for connections
	`time = seconds` (default: "**5**")	Idle time (after all connection channels are closed) allowed for a connection before automatically closing the connection

Element	Attributes and their values	Description
tcp-connect-timeout	`time = seconds` (default: "**5**")	Timeout for TCP connections (after which connection attempts to a Secure Shell server are stopped if the remote host is down or unreachable)
keepalive-interval	`time = seconds` (default: "**0**")	Time interval for sending keepalive messages to the Secure Shell server
exclusive-connection	`enable = "yes|no"`	A new connection is opened for each new channel
server-banners	`visible = "yes|no"`	Show server banner message file (if it exists) to the user before login
forwards / **forward**	`type = "x11|agent"`	Forwarding type
	`state = "on|off|denied"`	Set forwarding on or off, or deny it
remote-environment / **environment**	`name = env_var_name`	Name of an environment variable that is to be passed to the server from the client side
	`value = string`	Value of the environment variable
	`format = "yes|no"`	The Connection Broker processes Tectia-specific special variables in `value` (e.g. `%U%`)
server-authentication-methods / **auth-server-certificate**	-	Use certificates for server authentication
server-authentication-methods / **auth-server-publickey**	-	Use public host keys for server authentication
	`policy = "strict|ask|tofu|advisory"`	Policy for handling unknown server host keys
authentication-success-message	`enable = "yes|no"`	Output and log the `AuthenticationSuccessMsg` message
	`delay =seconds` (default: "**2**")	Time for how long authentication success message is shown
disconnect-message	`message = string`	Sets the message that displays when disconnecting.
keyboard-interactive	`prefix = string`	A message shown before of the keyboard-interactive prompt.
sftpg3-mode	`compatibility-mode = "tectia|ftp|openssh"`	Behavior of **sftpg3** when transferring files
terminal-selection	`selection-type = "select-words|select-paths"`	Behavior of the Tectia terminal when the user selects text with double-clicks

Element	Attributes and their values	Description
terminal-bell	`bell-style = "none|pc-speaker|`**`system-default`**`"`	Tectia terminal repeats audible notifications from destination (Unix) server
close-window-on-disconnect	`enable = "yes|`**`no`**`"`	Tectia GUI tab is to be closed while disconnecting from a server session by pressing **CTRL+D**
quiet-mode	`enable = "yes|`**`no`**`"`	Make **scpg3**, **sshg3**, and **sftpg3** suppress warnings, error messages and authentication success messages
checksum	`type = "`**`yes`**`|no|md5|sha1|md5-force|sha1-force|checkpoint"`	Default setting for comparing checksums
address-family	`type = "`**`any`**`|inet|inet6"`	IP address family: both, IPv4, or IPv6

Table A.5. `ssh-broker-config.xml` Quick Reference - the `profiles` element

Element	Attributes and their values	Description
profile	`id =` *`ID`*	Unique identifier that does not change during the lifetime of the profile
	`name =` *`string`*	Unique name (free-form text string) that can be used for connecting with the profile on the command line
	`host =` *`IP_address|FQDN|short_hostname`*	Secure Shell server host address
	`port =` *`port_number`* (default: "**`22`**")	Secure Shell server listener port number
	`protocol = "secsh2"`	The communications protocol used by the profile
	`host-type = "`**`default`**`|windows|unix"`	Server type for ASCII (text) file transfer
	`connect-on-startup = "yes|`**`no`**`"`	Connect automatically with the profile when the Connection Broker is started
	`user =` *`user_name`*	User name for opening the connection
	`gateway-profile =` *`profile_name`*	Create nested tunnels
profile / **hostkey**	`file =` *`path`*	Path to the remote server host public key file
profile / **ciphers** / **cipher**	`name =` *`cipher_name`*	A cipher used with this profile

Element	Attributes and their values	Description
profile / **macs** / **mac**	`name = MAC_name`	A MAC used with this profile
profile / **kexs** / **kex**	`name = KEX_name`	A KEX used with this profile
profile / **hostkey-algorithms** / **hostkey-algorithm**	`name = hostkey-algorithm_name`	Host key signature algorithm used with this profile
profile / **rekey**	`bytes = number` (default: `"1000000000"` (1 GB))	Number of transferred bytes after which key exchange is done again when using this profile
profile / **authentication-methods**	Define the authentication methods for this profile using the same child elements as with `default-settings / authentication-methods` (see Table A.4)	
profile / **user-identities** / **identity**	`file = path`	Path to the public-key file (primarily) or to a certificate
	`hash = hash`	Hash of the public key that will be used to identify the related private key
	`identity-file = path`	Reserved for future use
profile / **compression**	`name = "none|zlib"`	Compression settings (for the data that the client sends) used with this profile
	`level = [0 to 9]` (default: `"0"` (= level 6))	For `zlib`, compression level.
profile / **proxy**	`ruleset = rule_sequence`	Rules for HTTP proxy or SOCKS servers the client will use for connections with this profile
profile / **idle-timeout**	`type = "connection"`	Idle timeout is always defined for connections
	`time = seconds` (default: `"5"`)	Idle time (after all connection channels are closed) allowed for a connection before automatically closing the connection opened with this profile
profile / **tcp-connect-timeout**	`time = seconds` (default: `"5"`)	Timeout for TCP connections with this profile: Connection attempts to a Secure Shell server are stopped after the defined time if the remote host is down or unreachable
profile / **keepalive-interval**	`time = seconds` (default: `"0"`)	Time interval for sending keepalive messages to the Secure Shell server with this profile
profile / **exclusive-connection**	`enable = "yes|no"`	A new connection is opened for each new channel with this profile

Element	Attributes and their values	Description
profile / **server-banners**	`visible = "`**`yes`**`|no"`	Show server banner message file (if it exists) to the user before login with this profile
profile / **forwards** / **forward**	`type = "x11|agent"`	Forwarding type for this profile
	`state = "on|off|denied"`	Set forwarding on, off, or deny it (i.e. the user cannot enable it on the command-line) with this profile
profile / **tunnels** / **local-tunnel**	`type = "`**`tcp`**`|ftp|socks"`	Type of the local tunnel that is opened automatically when a connection is made with this profile
	`listen-address =` *`IP_address`* (default: **`127.0.0.1`**)	The network interfaces that should be listened on the client
	`listen-port =` *`port_number`*	Listener port number on the local client
	`dst-host =` *`IP_address|domain_name`* (default: **`127.0.0.1`**)	Destination host address
	`dst-port =` *`port_number`*	Destination port
	`allow-relay = "yes|`**`no`**`"`	Allow connections to the listened port from outside the client host
profile / **tunnels** / **remote-tunnel**	`type = "`**`tcp`**`|ftp"`	Type of the remote tunnel that is opened automatically when a connection is made with this profile
	`listen-address =` *`IP_address`* (default: **`127.0.0.1`**)	The network interfaces that should be listened on the server
	`listen-port =` *`port_number`*	Listener port number on the remote server
	`dst-host =` *`IP_address|domain_name`* (default: **`127.0.0.1`**)	Destination host address
	`dst-port =` *`port_number`*	Destination port
	`allow-relay = "yes|`**`no`**`"`	Allow connections to the listener port from outside the server host
profile / **remote-environment** / **environment**	`name =` *`env_var_name`*	Name of an environment variable that is to be passed to the server from the client side
	`value =` *`string`*	Value of the environment variable
	`format = "yes|`**`no`**`"`	The Connection Broker processes Tectia-specific special variables in `value` (e.g. `%U%`)

Element	Attributes and their values	Description
profile / **server-authentication-methods**	Define the server authentication methods allowed with this profile using the same child elements as with `default-settings / server-authentication-methods` (see Table A.4)	
profile / **password**	`string = password`	User password that the client will send as a response to password authentication
	`file = password_file`	File containing the password
	`command = path`	Path to a program or script that outputs the password

Table A.6. `ssh-broker-config.xml` Quick Reference - the `static-tunnels`, `gui`, and `logging` elements

Element	Attributes and their values	Description
static-tunnels / **tunnel**	`type = "tcp|ftp"`	Type of the static tunnel
	`listen-address = IP_address` (default: **127.0.0.1**)	The network interfaces that should be listened on the client
	`listen-port = port_number`	Listener port number on the local client
	`dst-host = IP_address|domain_name` (default: **127.0.0.1**)	Destination host address
	`dst-port = port_number`	Destination port
	`allow-relay = "yes|no"`	Allow connections to the listened port from outside the client host
	`profile = ID`	Connection profile ID that is used for the tunnel
gui	`hide-tray-icon = "yes|no"`	Hide the Tectia icon in the Windows taskbar notification area
	`show-exit-button = "yes|no"`	Show the **Exit** command in the Tectia icon's shortcut menu
	`show-admin = "yes|no"`	Show the **Configuration** command in the Tectia icon's shortcut menu
logging / **log-target**	`file = path`	File where the audit data is written to
	`type = "file|syslog|discard"`	Logging facility to which audit data is output
logging / **log-events**	`facility = "normal|daemon|user|auth|local0|local1|local2|local3|local4|local5|local6|local7|discard"`	Facility of logging event

Element	Attributes and their values	Description
	(On Windows: `facility = "normal|discard"`)	
	`severity = "informational |notice|warning|error| critical|security-success |security-failure"`	Severity of logging event
logging / **log-events** / **log-target**	The same as `logging / log-target`	

A.5 PrivX Desktop Shortcut Menu (Windows and Linux)

When Tectia Client (or Connection Broker) is running on Windows or Linux, and if so enabled in the operating system startup settings, the tray icon  is displayed in the notification area of the Windows taskbar, typically next to the time display at the bottom of the desktop.

Click the PrivX Desktop icon to open the GUI client.

Right-click the PrivX Desktop icon to open the shortcut menu.

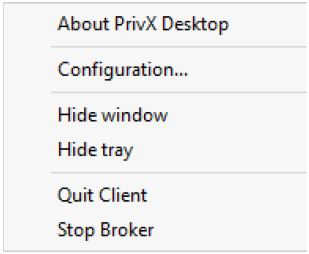


Figure A.39. The shortcut menu of PrivX Desktop

The menu has the following product-specific options:

- **About** shows the installed version and license information.
- **Configuration** opens the Tectia Connections Configuration GUI.
- **Quit Client** closes the PrivX Desktop GUI.
- **Stop Broker** stops the Connection Broker and closes all open connections.

Appendix B Command-Line Tools and Man Pages

Tectia Client is shipped with several command-line tools. Their functionality is briefly explained in the following appendices.

On Unix, the same information is available on the following manual pages:

- ssh-broker-g3(1): Connection Broker – Generation 3
- ssh-broker-ctl(1): Connection Broker control utility
- ssh-troubleshoot(8): utility for collecting system information for troubleshooting purposes
- sshg3(1): Secure Shell terminal client – Generation 3
- scpg3(1): Secure Shell file copy client – Generation 3
- sftpg3(1): Secure Shell file transfer client – Generation 3
- ssh-translation-table(1): Secure Shell file transfer translation table
- ssh-keygen-g3(1): authentication key pair generator
- ssh-keyfetch(1): utility for downloading server host keys
- ssh-cmpclient-g3(1): certificate CMP enrollment client
- ssh-scepclient-g3(1): certificate SCEP enrollment client
- ssh-certview-g3(1): certificate viewer
- ssh-ekview-g3(1): external key viewer

For a description of the Connection Broker configuration file options, see ssh-broker-config(5).

ssh-broker-g3

ssh-broker-g3 — Tectia Connection Broker - Generation 3

Synopsis

`ssh-broker-g3 [ -a, --broker-address=` *`ADDR`* `] [ -f, --config-file=` *`FILE`* `] [ -D, --debug=` *`LEVEL`* `] [ -l, --debug-log-file=` *`FILE`* `] [ --pid-file=` *`FILE`* `] [ --exit ] [ --reconfig ] [ -h ] [ -V ]`

Description

ssh-broker-g3 (**ssh-broker-g3.exe** on Windows) is a component of Tectia Client. It handles all cryptographic operations and authentication-related tasks for Tectia Client and for the client programs **sshg3**, **scpg3**, **sftpg3**, and **ssh-client-g3.exe** (on Windows only).

ssh-broker-g3 uses the Secure Shell version 2 protocol to communicate with a Secure Shell server.

You can start the Connection Broker manually by using the **ssh-broker-g3** command. This starts **ssh-broker-g3** in the background and all following uses of **sshg3**, **sftpg3**, or **scpg3** will connect via this instance of the Connection Broker instead of starting a new Broker session.

If a command-line client (**sshg3**, **sftpg3**, or **scpg3**) is started when the Connection Broker is not running in the background, the client starts the Broker in *run-on-demand* mode. In this mode, **ssh-broker-g3** will exit after the last client has disconnected.

If there is an **ssh-broker-g3** process running in the run-on-demand mode, and the Connection Broker is started from the command line, the new **ssh-broker-g3** process sends a message to the old **ssh-broker-g3** process to change from the run-on-demand mode to the background mode, keeping the Broker running after the clients disconnect.

The status of the running Connection Broker can be checked using the **ssh-broker-ctl** and **ssh-broker-gui** utilities.

Authentication

The Connection Broker operates automatically as an authentication agent, storing user's public keys and forwarding the authentication over Secure Shell connections. Key pairs can be created with `ssh-keygen-g3`.

The Connection Broker can also serve OpenSSH clients as an authentication agent.

The public key pairs used for user authentication are by default stored in the `$HOME/.ssh2` directory (`%APPDATA%\SSH\UserKeys` on Windows). See the section called “Files” for more information.

The Connection Broker automatically maintains and checks a database containing the public host keys used for authenticating Secure Shell servers. When logging in to a server host for the first time, the host's public key is stored in the user's `$HOME/.ssh2/hostkeys` directory (`%APPDATA%\SSH\HostKeys` on Windows). See the section called “Files” for more information.

Options

The most important options of **ssh-broker-g3** are the following:

`-a, --broker-address=` *`ADDR`*

Listens to Connection Broker connections on a local address *`ADDR`*.

`-D, --debug=` *`LEVEL`*

Sets the debug level string to *`LEVEL`*.

`-f, --config-file=` *`FILE`*

Reads the Connection Broker configuration file from *`FILE`* instead of the default location.

`-l, --debug-log-file=` *`FILE`*

Dumps debug messages to *`FILE`*.

`--pid-file=` *`FILE`*

Stores the process ID of the Connection Broker to *`FILE`*.

`--exit`

Make the currently running Connection Broker exit. This will terminate all connections.

`--reconfig`

Re-reads the configuration file (`ssh-broker-config.xml`) and takes it into use.

`-V, --version`

Displays program version and exits.

`-h, --help`

Displays a short summary of command-line options and exits.

Environment Variables

The following optional environment variables are required in certain situations:

SSH_SECSH_BROKER =*`ADDRESS`*

This variable defines an address to a separate Tectia Connection Broker process to which a connection is made.

This variable becomes necessary to define the location of the Connection Broker process, if you are running it from a non-default location, or using a userID other than that of the **ssh-broker-g3** process owner.

Files

ssh-broker-g3 uses the following files:

`$HOME/.ssh2/ssh-broker-config.xml`

This is the user-specific configuration file used by **ssh-broker-g3** (and **sshg3**, **scpg3**, and **sftpg3**). The format of this file is described in ssh-broker-config(5). This file does not usually contain any sensitive information, but the recommended permissions are *read/write* for the user, and *not accessible* for others.

On Windows, the user-specific configuration file is located in `%APPDATA%\SSH\ssh-broker-config.xml`.

`$HOME/.ssh2/random_seed`

This file is used for seeding the random number generator. It contains sensitive data and its permissions should be *read/write* for the user and *not accessible* for others. This file is created the first time the program is run and it is updated automatically. You should never need to read or modify this file.

On Windows, the random seed file is located in `%APPDATA%\SSH\random_seed`.

`$HOME/.ssh2/identification`

This file contains information on public keys and certificates used for user authentication when connecting to remote hosts.

With Tectia Client G3, using the `identification` file is not necessary if all user keys are stored in the default directory and you allow all of them to be used for public-key and/or certificate authentication. If the `identification` file does not exist, the Connection Broker attempts to use each key found in the `$HOME/.ssh2` directory. If the `identification` file exists, the keys listed in it are attempted first.

The identification file contains a list of private key filenames each preceded by the keyword `IdKey` (or `CertKey`). An example file is shown below:

```
IdKey       mykey
```

This directs the Connection Broker to use `$HOME/.ssh2/mykey` when attempting login using public-key authentication.

The files are by default assumed to be in the `$HOME/.ssh2` directory, but also a path to the key file can be given. The path can be absolute or relative to the `$HOME/.ssh2` directory. If there is more than one `IdKey`, they are tried in the order that they appear in the identification file.

On Windows, the identification file is located in `%APPDATA%\SSH\identification`. Key paths in the file can be absolute or relative to the `%APPDATA%\SSH` directory and include same pattern strings as supported for authorization file on server-side, for example `C:\%username-without-domain%\private_keys\mykey`. The default user key directory is `%APPDATA%\SSH\UserKeys` and the default user certificate directory is `%APPDATA%\SSH\UserCertificates`.

`$HOME/.ssh2/hostkeys`

This is the user-specific default directory for storing the public keys of server hosts. You are prompted to accept new or changed keys automatically when you connect to a server, unless you have set `strict-host-key-checking` to `yes` in the `ssh-broker-config.xml` file. You should verify the key fingerprint before accepting new or changed keys.

When the host key is received during the first connection to a remote host (or when the host key has changed) and you choose to save the key, its filename is stored by default in hashed format. The hashed host key format is a security feature to make address harvesting on the hosts difficult.

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

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

If both hashed and plain-text format keys exist, the hashed format takes precedence.

Note that the identification is different based on the host and port the client is connecting to. For example, the short hostname *`alpha`* is considered different from the fully qualified domain name *`alpha.example.com`*. Also a connection with an IP, for example *`10.1.54.1`*, is considered a different host, as is a connection to the same host but different port, for example *`alpha.example.com#222`*.

On Windows, the user-specific host key files are located in `%APPDATA%\SSH\HostKeys`.

For more information on host keys, see Section 4.2.

`$HOME/.ssh2/hostkeys/salt`

This is the initialization file for hashed host key names.

On Windows, the salt file is located in `%APPDATA%\SSH\HostKeys\salt`.

`/opt/tectia/share/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`

This is the configuration file used by **ssh-broker-g3** (and **sshg3**, **scpg3**, and **sftpg3**) that contains the factory default settings. It is not recommended to edit the file, but you can use it to view the default settings. The format of this file is described in ssh-broker-config(5).

On Windows, the default configuration file is located in `<INSTALLDIR>\SSH Tectia AUX\ssh-broker-ng\ssh-broker-config-default.xml`.

`/etc/ssh2/ssh-broker-config.xml`

This is the global (system-wide) configuration file used by **ssh-broker-g3** (and **sshg3**, **scpg3**, and **sftpg3**). The format of this file is described in ssh-broker-config(5).

On Windows, the global configuration file is located in `<INSTALLDIR>\SSH Tectia Broker\ssh-broker-config.xml`.

`/etc/ssh2/hostkeys`

If a host key is not found in the user-specific `$HOME/.ssh2/hostkeys` directory, this is the next location to be checked for all users. Host key files are not automatically put here but they have to be updated manually by the system administrator (`root`).

If the administrator obtains the host keys by connecting to each host, the keys will be by default in the hashed format. In this case, also the administrator's `$HOME/.ssh2/hostkeys/salt` file has to be copied to the `/etc/ssh2/hostkeys` directory.

On Windows, the system-wide host key files are by default located in:

`"C:\ProgramData\SSH\HostKeys"`

`/etc/ssh2/hostkeys/salt`

This is the initialization file for hashed host key names. The file has to be copied here manually by the same administrator that obtains the host keys.

On Windows, the salt file for all users is by default located in:

`"C:\ProgramData\SSH\HostKeys\salt"`

`/etc/ssh/ssh_known_hosts`

This is the default system-wide file used by OpenSSH clients for storing the public key data of known server hosts. It is supported also by Tectia Client.

If a host key is not found in the user-specific `$HOME/.ssh/known_hosts` file, this is the next location to be checked for all users.

The `ssh_known_hosts` file is never automatically updated by Tectia Client, since they store new host keys always in the Tectia user-specific directory `$HOME/.ssh2/hostkeys`.

`$HOME/.ssh/known_hosts`

This is the default user-specific file used by OpenSSH clients for storing the public key data of known server hosts. The `known_hosts` file is supported also by Tectia Client.

The `known_hosts` file contains a hashed or plain-text format entry of each known host key and the port used on the server, in case it is non-standard (other than 22). For more information on the format of the `known_hosts` file, see the OpenSSH sshd(8) man page.

The `known_hosts` file is never automatically updated by Tectia Client, since they store new host keys always in the Tectia directory `$HOME/.ssh2/hostkeys`.

`$HOME/.ssh2/authorized_keys` (on the server host)

This directory is the default location used by Tectia Server for the user public keys that are authorized for login.

On Tectia Server on Windows, the default directory for user public keys is `%USERPROFILE%\.ssh2\authorized_keys`.

`$HOME/.ssh2/authorization` (on the server host)

This is the default file used by earlier versions of Tectia Server (**sshd2**) that lists the user public keys that are authorized for login. The file can optionally be used with Tectia Server G3 (**ssh-server-g3**) as well.

On Tectia Server on Windows, the authorization file is by default located in `%USERPROFILE%\.ssh2\authorization`.

For information on the format of this file, see the ssh-server-g3(8) man page.

`$HOME/.ssh/authorized_keys` (on the server host)

This is the default file used by OpenSSH server (**sshd**) that contains the user public keys that are authorized for login.

For information on the format of this file, see the OpenSSH sshd(8) man page.

ssh-broker-ctl

ssh-broker-ctl — Tectia Connection Broker control utility

Synopsis

`ssh-broker-ctl` *command*
[`options` ...]

Description

ssh-broker-ctl (**ssh-broker-ctl.exe** on Windows) is a control utility for the Connection Broker (**ssh-broker-g3**). It can be used, for example, to view the status of the Connection Broker, to reconfigure or stop the Connection Broker, to manage keys and certificates, and to manage connections.

Options

The following general options are available:

`-a, --broker-address` *ADDRESS*

Defines an address to a separate Tectia Connection Broker process to which a connection is made.

The same effect can be achieved by defining a Connection Broker address with environment variable `SSH_SECSH_BROKER`.

Tip

If you are running **ssh-broker-ctl** using a userID other than that of the **ssh-broker-g3** process owner, the `-a` option must be given so that **ssh-broker-ctl** knows where to connect. In this case, you must also run **ssh-broker-ctl** as a privileged user (root).

For example, when user *SSHBRKR* owns the **ssh-broker-g3** process, run the **ssh-broker-ctl** with commands:

```
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker status -s
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker status --pid
# ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker list-connections
```

`-D, --debug` *STR*

Defines the debug level.

`-e, --charset=` *CS*

Defines the character set to be used in the output. The supported character sets are `utf8`, `iso-8895-1`, `latin1`, `iso-8859-15`, `latin9`, and `ascii`.

`-q, --quiet`

Defines that little or no output is to be displayed, depending on the command.

`-s, --short`

Defines that a shorter, more machine readable, output format is to be used.

`--time-format=` *`FMT`*

Defines the time format to be used in the output. The default depends on the system locale settings.

`-v, --verbose`

Defines that more information, if available, is to be output.

`-V, --version`

Displays the version string.

`-w, --wide`

Defines that the output will not be truncated, even if it means long lines.

`-h, --help`

Displays a context-sensitive help text on command-line options. Help is available also on specific commands. For example, to get help on the `status` command, run:

```
$ ssh-broker-ctl status --help
```

Commands

Note

For a detailed description of the command options, use the command-specific `--help` option.

ssh-broker-ctl accepts the following commands:

`add-certificate` [`options`] *`<certificate-file>`*

Adds the given X.509 sub-CA certificate to the Connection Broker certificate cache. The certificate can be used in certificate validations but it is not stored permanently. Restarting the Connection Broker will remove the certificate.

`add-crl` [`options`] *`<crl-file>`*

Adds the given X.509 CRL to the Connection Broker CRL cache. The CRL can be used in certificate validations but it is not stored permanently. Restarting the Connection Broker will remove the CRL.

`add-key` *`filename`*

Adds a new private key from the given file name. The private key is not stored permanently in the configuration. Stopping the Connection Broker will remove the key.

`add-provider` *`type parameter`*

Registers a key provider to the Connection Broker. The `type` option is one of the supported provider types and the `parameter` option is a parameter string specific to the provider type.

For a list of the supported key provider types and the corresponding parameter formats, use the command-specific `--help` option.

`auth-handler` [`options`]

Registers itself as the default authentication form handler. All authentication prompts for clients that are unable to handle them (mostly SOCKS proxy and other tunnels) are directed to this client.

For a list of the supported key provider types and the corresponding parameter formats, use the command-specific `--help` option.

`capture-driver` [`install [--path="PATH"|--disable-auto-start]`] [`uninstall [--force]`] [`start`] [`stop [--force]`] [`reconfig`] [`debug <level> [--save-to="PATH"|--append|--clear]`]

Manage the Windows capture component (driver).

You can install the driver with the `install` command. The `install` command may optionally take a custom installation path with `--path`. You may also disable automatic driver start with `--disable-auto-start`. Note that the driver is automatically installed when you installed Tectia Client with the Transparent TCP Tunneling feature.

The `uninstall` command may optionally use `--force`, in which case a Stop signal is sent even if the driver can't stop instantly.

When you `stop` the driver, existing tunnels aren't affected. This only stops filtering new connections. Use `--force` to send the Stop signal even if the driver cannot stop instantly.

The `reconfig` reads the capture configuration from the global configuration file and then writes it to the registry for the driver to read. The driver will hot-reload the configuration. All the connected brokers will reconnect to the driver. Running `reconfig` always restarts the trusted broker if it is enabled and set to run on boot. This also means stopping any broker that is using the socket address 'capture-trusted-broker'. The configuration is written to `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SSHCaptureDriver\Parameters\Configuration`.

You can set `debug` to 0 or greater to obtain driver logs. The `debug` command also supports the following options:

`--save-to="PATH"` for saving the debug logs to the specified path, if the path ends with .evtx logs are saved in EventLog format.
`--append` to append new log entries to the file (instead of overriding).
`--clear` sets debug level to 0 and disables the Debug log.

The following levels are available for `debug`:

0 - Log nothing.
1 - Log critical errors.
2 - Log regular errors.
3 - Log warnings.
4 - Log important events and high-level successes.
5+ - Verbose debug.

`close-channel` *`channel-id ...`*

Closes the defined channel. You can also enter multiple channel-IDs to close several channels.

`close-connection` *`connection-id ...`*

Closes the defined connection. You can also enter multiple connection-IDs to close several connections.

`close-tunnel-listener` *`tunnel-id ...`*

Closes open tunnel listener. Tunnel id is either the id number returned by **ssh-broker-ctl list-tunnel-listeners** command or a listen address and port pair separated by a colon. If the listen address is omitted, local listeners (127.0.0.1) are selected. As an example, the following command closes the listener with id 7, and the ones listening at 168.192.0.15 port 1234 and 127.0.0.1 port 2112:

```
$ ssh-broker-ctl ctl 7 168.192.0.15:1234 :2112
```

`config-value` [`options`] *`path`*

Retrieves configuration values from the Connection Broker based on the defined path and displays them.

`connection-status` [`--show-channels`] [`--write-hostkey=` *`FILE`*] *`connection-id`*

Displays a detailed connection status for the connection ID (the numeric identifier shown by the **list-connections**) command.

`disconnect-client` *`client-id`*

Disconnects a Connection Broker client process.

`debug` [`--append`] [`--clear`] [`--log-file=` *`file`*] [`--monitor`] [`--protocol-dump`] [*`debug-level`*]

Sets the Connection Broker debug level to the defined level. If no `debug-level` parameter is given here, the current debug level is not changed.

`generate-key` [`options`] *`key-name`*

Generates a private key using a key provider in the Connection Broker. By default the private key will be stored as a software key into a file in the user's home directory. Key providers can offer other methods for private key storage.

`keylog` [`--remove`] [`--all`] [`--update` <*key-id*|*key-hash*>] [`--init`] [`--uninit`] [`--close`] [`-v, --verbose`] [`key-id|key-hash|hostname`]

Keylog is used to manage uploaded public keys and to display a log of them. The Keylog does not store the public keys, it only stores information about the keys and the hosts where the keys have been uploaded to. The information can be used to manage the keys at a later stage, for example, to track hosts where a key has been uploaded to. The keylog is not on by default, it must be enabled first.

Without the options, displays a list of the uploaded keys. If a key or a hostname is specified, only the selected keys are displayed.

`key-passphrase` [`--all`] [`--clear`] [`--passphrase-file=` *filename*] [`--passphrase-string=` *passphrase*] [*key-id*|*key-hash*]

Prompts the user's private-key passphrase or PIN code.

`key-upload` [`options`] [`--replace-key`] [`--scan-key`] [`--delete-key`] *key* [*user*@]*server* [*#port*]

Uploads the selected key (`key` can be a key ID number, a public key hash or a file name) into the authorized keys directory or file on the server, depending on the automatically detected upload method. After the operation, the key can be used in public-key authencation to log in to the server without a password. If the keylog is enabled, the command prompts for a keylog passphrase (if needed), and information about the public keys is stored in the key upload log.

The option `--replace-key` will rotate the selected key according to normal key rotation rules. The option `--scan-key` can be used to scan the selected hosts' keys. The option `--delete-key` can be used to delete the selected authorized key(s).

`list-connections` [`-c, --show-channels`] [`-s, --short`] [`--client-pid=` *PID*] [`--disconnected`]

Displays a list of the currently open connections together with connection parameters and traffic statistics. Displays also the connection ID which can used with other commands to identify the connection.

`list-channels` [`-s, --short`]

Displays a list of the currently open connection channels, together with channel type and traffic statistics. Displays also the channel ID which is used by other commands to identify the connection.

`list-clients` [`-c, --show-channels`] [`-s, --short`] [`--all`]

Displays a list of the currently connected client processes.

`list-keys` [`-s, --short`] [`--extra-certificates`] [`--provider=` *ID*]

Displays a list of the user's private keys, together with the basic key attributes such as the key type, size, and possible file name or key provider information. Outputs also the fingerprint and the identifier of the key. The identifier is used by other Connection Broker commands to identify the private key.

Use `--short` option to display a one-line description per user private key. Option `--extra-certificates` lists extra certificates (CAs, certificates without private key) instead of user keys and certificates. Option `--provider=ID` lists only keys reported by a given provider. The ID can be either a number or a provider name.

`list-profiles` [`-s, --short`] [`-v, --verbose`] [*`name ...`*]

Displays a list of connection profiles in the Connection Broker. Shows the profile name and basic connection settings, such as the host and the user name. If profile names are given, only those profiles are listed.

`list-providers` [*`provider ...`*]

Displays a list of the key providers in the Connection Broker. If one or more provider names or ID numbers are given, only those providers will be listed. The provider name can be either a full provider name or a prefix.

`list-tunnel-listeners` [`options`]

Displays a list of the currently active tunnel listeners (also called port forwards).

`open-tunnel-listener` [`options`] *`listen-port`* [*`user@`*]*`server`* [*`#port`*] [*`dst-host`*] [*`dst-port`*]

Opens a tunnel listener, similar to **sshg3** `-L` and `-R` options. The difference is that **ssh-broker-ctl** will exit after the tunnel is opened. The tunnel status can be viewed with **ssh-broker-ctl list-tunnel-listeners** command and the tunnel can be closed with **ssh-broker-ctl close-tunnel-listener** command.

In local mode (default), the listener is opened to localhost listen-port. All connections will be tunneled through server and from there to the final destination address and port. Tunnel types `socks` and `socks-proxy` do not require destination information as it will be obtained from SOCKS client. Tunnel types `tcp`, `ftp` and `local` require destination information.

`performance` [`options`] [`show`] [`clear`] [`show-and-clear`] [`show-total`] [`interval` <*`time`*|*`index`*|*`'all'`*>]

Show and manipulate performance profiling data.

`pkcs10-sign` [`options`] *`key-id`* [*`subject-name`*]

Signs a PKCS#10 certificate request with the given key. The `key-id` can be either a key id or a key hash. The subject name parameter is required unless the `template` option is used. If the subject name is not a valid distinguished name, it will be wrapped automatically into a common name component. For example, a subject name string `My Name` will be converted to `CN=My Name`.

`probe-key` [`options`] *`address#port`*

Probes for a Secure Shell server hostkey. Connects to the given address and port (defaults to 22) and displays the server's public key or certificate.

`reload`

Rereads the Connection Broker configuration file.

`remove-key` [`options`] *`key-id`*

Removes a private key permanently.

`remove-provider` [`--all`] *`provider-id`*

Removes a key provider from the Connection Broker.

`start`

Starts the Connection Broker in daemon mode if it is not already running.

`start-gui`

Starts the Connection Broker GUI process unless it is already running.

`status` [`--all`] [`--defaults`] [`--pid`] [`-q, --quiet`] [`-s, --short`]

Without parameters, displays short statistics and a configuration summary for the currently running Connection Broker process.

With `--defaults`, the command lists the default settings instead of all supported algorithms.

`stop`

Stops the Connection Broker.

`validate-certificate` [`options`] <*`certificate-file`*>

Validates the given X.509 certificate. If a host name is given, also checks if the certificate would be accepted as a host certificate for the host.

`view-key` [`-s, --short`] [`-v, --verbose`] [`--clear`] [`--write-key=` *`file`*] *`key-id`*

Displays information on the defined key. If the key has certificates, a short summary of them is also shown.

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.

sshg3

sshg3 — Secure Shell terminal client - Generation 3

Synopsis

`sshg3` [`options` ...]
`profile` | [*`user@`*] *`host`* [*`#port`*]
[*`command`*]

Description

sshg3 (**sshg3.exe** on Windows) is a program for logging in to a remote machine and executing commands on a remote machine. **sshg3** provides secure, encrypted communication channels between two hosts over an unsecured network. It can be used to replace the unsecured **rlogin**, **rsh**, and **telnet** programs. Also X11 connections and arbitrary TCP/IP ports can be forwarded over secure channels with **sshg3**.

To connect to a remote host using **sshg3**, give either the name of a connection profile defined in the `ssh-broker-config.xml` file (*`profile`*) or the IP address or DNS name of the remote host, optionally with the remote user name and the port of the Secure Shell server (*`[user@]host[#port]`*). If no user name is given, the local user name is assumed. If no port is given, the default Secure Shell port 22 is assumed. The remote host must be running a Secure Shell version 2 server.

sshg3 acts as a Connection Broker client and launches the actual Connection Broker process, **ssh-broker-g3** as a transport (in run-on-demand mode), or uses an already running Connection Broker process. The Connection Broker will ask the user to enter a password or a passphrase if they are needed for authentication. Connection Broker uses the configuration specified in the `ssh-broker-config.xml` file.

When the user's identity has been accepted by the server, the server either executes the given command, or logs in to the machine and gives the user a normal shell. All communication with the remote command or shell will be automatically encrypted.

If no pseudo-tty has been allocated, the session is transparent and can be used to securely transfer binary data.

The session terminates when the command or shell on the remote machine exits and all X11 and TCP/IP connections have been closed. The exit status of the remote program is returned as the exit status of **sshg3**.

Agent Forwarding (Unix)

ssh-broker-g3 acts as an authentication agent, and the connection to the agent is automatically forwarded to the remote side unless disabled in the `ssh-broker-config.xml` file or on the **sshg3** command line (with the `-a` option).

X11 Forwarding

If the user is using X11 (the `DISPLAY` environment variable is set), the connection to the X11 display can be automatically forwarded to the remote side in such a way that any X11 programs started from the

shell (or command) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set `DISPLAY`. X11 connection forwarding can be allowed in the `ssh-broker-config.xml` file or on the **sshg3** command line (with the `+x` option). By default, X11 forwarding is disabled.

The `DISPLAY` value set by **sshg3** will point to the server machine, but with a display number greater than zero. This is normal, and happens because **sshg3** creates a "proxy" X server on the server machine for forwarding the connections over the encrypted channel.

sshg3 will also automatically set up the Xauthority data on the server machine. For this purpose, it will generate a random authentication cookie, store it in the Xauthority data on the server, and verify that any forwarded connections carry this cookie and replace it with the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the plain).

TCP Port Forwarding

Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either in the `ssh-broker-config.xml` file or on the **sshg3** command line (with the `-L` and `-R` options).

Options

Command-line options override the settings in the `ssh-broker-config.xml` file if the same option has been configured in both places. The following options are available:

`-a, --no-agent-forwarding`

Disables authentication agent forwarding. In the factory settings, agent forwarding is enabled.

`+a`

Enables authentication agent forwarding. In the factory settings, agent forwarding is enabled, but it can be denied in the Connection Broker configuration file, in which case users cannot enable it on the command-line and this `+a` will be ignored.

`-B, --batch-mode`

Uses batch mode. Fails authentication if it requires user interaction on the terminal.

Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

`-C`

Disables compression from the current connection.

`+C`

Enables zlib compression for this particular connection.

`-c, --ciphers=` *`LIST`*

Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported cipher names.

`-D, --debug=` *`LEVEL`*

Sets the debug level. *`LEVEL`* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

Note

Option `-D` only applies on Unix. On Windows, instead of this command-line tool, use the Connection Broker debugging options `-D, -l`.

Note

The debug level can be set only when the **sshg3** command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

`-e, --escape-char=` *`CHAR`*

Sets escape character (none: disabled, default: ~).

`-f, --fork-into-background`

Forks into background mode after authentication (Unix only). Use this option with tunnels and remote commands. Implies `-S` (unless a command is specified). When tunnels have been specified, this option makes **sshg3** stay in the background, so that it will wait for connections indefinitely. **sshg3** has to be killed to stop listening.

`-g, --gateway`

Gateways ports, which means that also other hosts may connect to forwarded ports. This option has to be specified before the "`-L`" or "`-R`" option. Note the logic of `+` and `-` in this option.

`+g`

Does not gateway ports. Listens to tunneling connections originating only from the localhost. This is the default value. Note the logic of `+` and `-` in this option.

`-i` *`FILE`*

Defines that private keys defined in the identification file are used for public-key authentication.

`-K, --identity-key-file=` *`FILE`*

Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple `-K` options on command line.

`-L, --localfwd` [*`protocol/`*] [*`listen-address:`*] *`listen-port:dst-host:dst-port`*

Forwards a port on the local (client) host to a remote destination host and port.

This allocates a listener port (*`listen-port`*) on the local client. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the remote server and another connection is made from the server to a specified destination host and port (*`dst-host:dst-port`*). The connection from the server onwards will not be secure, it is a normal TCP connection.

Giving the argument *`protocol`* enables protocol-specific forwarding. The protocols implemented are `tcp` (default, no special processing), `ftp` (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session), and `socks`.

With the `socks` protocol, the syntax of the argument is "`-L` *`socks/[listen-address:]listen-port`*". When this is set, Tectia Client acts as a SOCKS server for other applications, creating forwards as requested by the SOCKS transaction. This supports both SOCKS4 and SOCKS5.

If *`listen-address`* is given, only that interface on the client is listened. If it is omitted, only local interface is listened unless the `--gateway` is used to bind to all interfaces on the client-side.

`-l, --user=` *`USERNAME`*

Logs in using this user name.

`-m, --macs=` *`LIST`*

Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

```
--macs hmac-sha1-96,hmac-md5,hmac-md5-96
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported MAC names.

`-u, --kexs=` *`kexs`*

Sets the allowed key exchange (KEX) methods to be offered to the server. List the KEX names in a comma-separated list. For example:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported KEX methods.

`-j, --hostkey-algs=` *`algs`*

Sets the allowed host key algorithms to be offered to the server. List the host key algorithms in a comma-separated list. For example:

```
--hostkey-algs ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported host key algorithms.

`-n, --dev-null` (Unix) , `-n, --null` (Windows)

Redirects input from `/dev/null` (Unix) and from `NUL` (Windows).

`-o` *`option`*

Processes an option as if it was read from a Tectia Client 4.x-style configuration file. The supported options are `ForwardX11`, `ForwardAgent`, `AllowedAuthentications` and `PidFile`. For example, `-o "ForwardX11=yes"`. Also `-o "PidFile=/tmp/sshg3.pid"` makes sshg3 to store its process ID into file "/tmp/sshg3.pid" if it goes into background.

`-P, --password=` *`PASSWORD`* | `file://` *`PASSWORDFILE`* | `extprog://` *`PROGRAM`*

Sets user password that the client will send as a response to password authentication. The *`PASSWORD`* can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (`--password=file://`*`PASSWORDFILE`*), or entering a path to a program or script that outputs the password (`--password=extprog://`*`PROGRAM`*).

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

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

Caution

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

`-p, --port=` *`PORT`*

Connects to this port on the remote host. A Secure Shell server must be listening on the same port.

`--publickey-algorithms=` *`PUBLICKEY_ALGORITHMS`*

Allow only selected signature algorithms to be used in public key authentication. For example:

```
--publickey-algorithms=x509v3-ssh-rsa,rsa-sha2-512
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the available signature algorithms.

`-q`

Quiet mode, reports only fatal errors. This option overrides the `quiet-mode` setting made in the Connection Broker configuration file.

`-R, --remotefwd` [*`protocol/`*] [*`listen-address:`*] *`listen-port:dst-host:dst-port`*

Forwards a port on the remote (server) host to a destination host and port on the local side.

This allocates a listener port (*`listen-port`*) on the remote server. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified destination host and port (*`dst-host:dst-port`*). The connection from the client onwards will not be secure, it is a normal TCP connection.

Giving the argument *`protocol`* enables protocol-specific forwarding. The protocols implemented are `tcp` (default, no special processing) and `ftp` (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session).

If *`listen-address`* is given, only that interface on the server is listened. If it is omitted, only local interface is listened unless the `--gateway` is used to bind to all interfaces on the server-side.

`-S, --no-session-channel`

Does not request a session channel. This can be used with port-forwarding requests if a session channel (and tty) is not needed, or the server does not give one.

`+S`

Requests a session channel. This is the default value.

`-s, --subsystem` *`subsystem remote_server`*

Sets a subsystem or a service to be invoked on the remote server. The subsystem is specified as a remote command. For example: `sshg3 -s sftp <server>`

`-t, --tty`

Allocates a tty even if a command is given.

`-v, --verbose`

Uses verbose mode. More information or error diagnostics are output if a connection fails.

`-x, -X, --no-x11-forwarding`

Disables X11 connection forwarding. This is the default value.

`+x, +X`

Enables X11 connection forwarding.

`-z, --broker-log-file=` *`FILE`*

Sets the Connection Broker log file to *`FILE`*. This option works only if **ssh-broker-g3** gets started by this process).

`--aa, --allowed-authentications=` *`METHODS`*

Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. For example:

```
--allowed-authentications keyboard-interactive,password
```

Enter `help` as the value to view the currently supported authentication methods.

`--abort-on-failing-tunnel`

Aborts if creating a tunnel listener fails (for example, if the port is already reserved).

`--any-alg`

Allow any supported cipher, mac, kex, hostkey and publickey algorithm to be used.

`--compressions=` *`METHODS`*

Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter `help` as the value to view the currently supported compression methods.

`--disconnect-message=`*`MESSAGE`*

Sets a message that is displayed when disconnected. The message may contain any of the following substitution variables:

- `time`: Time of disconnect.
- `random`: 16 random hexadecimal digits.
- `random4`: 4 random hexadecimal digits.
- `random8`: 8 random hexadecimal digits.
- `random16`: Same as *`random`*.
- `pid`: Process id of sshg3.
- `broker_pid`: Broker process id.
- `conn_id`: Connection id.
- `session_id`: Session id.

- `target_host`: Target server name.
- `target_port`: Target server port.

If any of the random variables are used in a disconnect message, the random values are displayed to the user before the authentication-success message.

`--exclusive`

Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

`--hostkey-policy=` *`POLICY`*

Defines the policy for checking server host keys and handling unknown server host keys. The possible values are:

- `ask` (default): The user will be asked to verify and accept the server host keys, if the keys are not found in the host key storage or if the keys have changed.
- `strict`: The connection to the server will be allowed only if the host key is found in the user's known host keys storage.
- `tofu`: Trust on first use; new host keys are stored without prompting the user to accept them.
- `advisory` (*not recommended*): New host keys are stored without prompting the user to accept them, and connections are allowed also to servers offering a changed host key.

Caution

Consider carefully before setting the policy to `advisory`. Disabling the host-key checks makes the connection vulnerable to attacks.

You can also configure the host key policy in the `ssh-broker-config.xml` configuration file with the `<auth-server-publickey>` element in the `default-settings` and per profile. See **auth-server-publickey** .

If this option is set on the command-line client and configured in the `ssh-broker-config.xml`, the command-line value will be used.

`--identity=` *`ID`*

Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

`--identity-key-hash` *`ID`*

Defines the private key used in user authentication with the corresponding public key hash.

`--identity-key-id` *`ID`*

Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

`--keep-alive=` *`VALUE`*

Defines how often keep-alive messages are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

`--kip`

Defines keyboard-interactive and password as the allowed methods for user authentication; the same as

```
--allowed-authentications keyboard-interactive,password
```

`--remote-environment name=` *`VALUE`*

When this option is used, the defined environment variables are passed to the server from the client side. The environment variables are applied on the server when requesting a command, shell or subsystem.

Note that the server can restrict the setting of environment variables.

You can also configure the environment variables to be passed to the server in the `ssh-broker-config.xml` configuration file with the `<remote-environment>` element in the `default-settings` and per profile. See **remote-environment** .

If the same variable is entered on the command-line client and configured in the `ssh-broker-config.xml`, the command-line version will be used.

`--remote-environment-format name=` *`VALUE`*

The defined environment variables are passed to the server from the client side. The Connection Broker processes the value before sending it to the server.

You can use `%U` in the `value` to indicate a user name. The Connection Broker replaces the `%U` with the actual user name before sending it to the server.

For more information, see the `--remote-environment` option above.

`--tcp-connect-timeout=` *`VALUE`*

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter the value as a positive number.

`--template-profile` *`profile`*

Use the specified profile when connecting.

This may be useful for example when connecting to legacy servers that do not support modern algorithms. In such cases you may define in `ssh-broker-config.xml` configuration file a profile with the required algorithms similar to the following.

```
<profile name="legacy"
```

```
        host=""
        id="legacy">
  <kexs>
    <kex name="diffie-hellman-group1-sha1" />
  </kexs>
  <hostkey-algorithms>
    <hostkey-algorithm name="ssh-dss" />
  </hostkey-algorithms>
  <ciphers>
    <cipher name="aes128-cbc" />
  </ciphers>
</profile>
```

Then connect with:

```
$ sshg3 --template-profile legacy user@host
```

CAUTION: Using insecure algorithms can lead to data leaks. We recommend you rather update your legacy servers to support modern algorithms.

`-V, --version`

Displays program version and exits.

`-h, --help, -?`

Displays a short summary of command-line options and exits.

Commands

sshg3 can take as a command either of the following ones:

`remote_command` *`[arguments] ...`*

Runs the command on a remote host.

`-s` *`service`*

Enables a service in remote server.

Escape Sequences

sshg3 supports escape sequences to manage a running session. For an escape sequence to take effect, it must be typed directly after a newline character (press **Enter** first). The escape sequences are not displayed on screen during typing.

The following escape sequences are supported:

~.

Terminates the connection.

~Ctrl-Z

Suspends the session.

~~

Sends the escape character literally.

~#

Lists forwarded connections.

~-

Disables the escape character irrevocably.

~?

Displays a summary of escape sequences.

~l

Enable linemode.

~r

Initiates rekeying manually.

~s

Gives connection statistics, including server and client version, packets in, packets out, compression, key exchange algorithms, public-key algorithms, and symmetric ciphers.

~u

Uploads the chosen public key automatically to the server. If the user has only one key, it will be uploaded. Otherwise the largest key with a name that matches `id_rsa_<size>_a` will be selected.

~U

Uploads a public key to the server. A list of available keys is printed and the user is prompted to select one to be uploaded.

~c

Gives statistics for individual channels (data window sizes etc). This is for debugging purposes.

~V

Dumps the client version number to stderr (useful for troubleshooting).

Environment Variables

Upon connection, the Secure Shell server will automatically set a number of environment variables that can be used by **sshg3**. The exact variables set depend on the Secure Shell server. The following variables can be used by **sshg3**:

`DISPLAY`

The `DISPLAY` variable indicates the location of the X11 server. It is automatically set by the server to point to a value of the form `hostname:n` where `hostname` indicates the host on which the server and the shell are running, and `n` is an integer greater than or equal to 1. **sshg3** uses this special value to forward X11 connections over the secure channel.

The user should normally not set `DISPLAY` explicitly, as that will render the X11 connection unsecured (and will require the user to manually copy any required authorization cookies).

`HOME`

The user's home directory.

`LOGNAME`

Synonym for `USER`; set for compatibility with systems using this variable.

`MAIL`

The user's mailbox.

`PATH`

Set to the default PATH, depending on the operating system or, on some systems, `/etc/environment` or `/etc/default/login`.

`SSH_SOCKS_SERVER`

The address of the SOCKS server used by **sshg3**.

`SSH2_AUTH_SOCK`

If this exists, it is used to indicate the path of a Unix-domain socket used to communicate with the authentication agent (or its local representative).

`SSH2_CLIENT`

Identifies the client end of the connection. The variable contains three space-separated values: client IP address, client port number, and server port number.

`SSH2_ORIGINAL_COMMAND`

This will be the original command given to **sshg3** if a forced command is run. It can be used, for example, to fetch arguments from the other end. This does not have to be a real command, it can be the name of a file, device, parameters or anything else.

`SSH2_TTY`

This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

`TZ`

The time-zone variable is set to indicate the present time zone if it was set when the server was started (the server passes the value to new connections).

`USER`

The name of the user.

For a list of varibles set by Tectia Server, see the ssh-server-g3(8) man page.

Exit Values

sshg3 returns the following values based on the result of the operation:

```
0     Operation was successful.
1     sshg3 has encountered an error,
      the reason is usually given in an error message.
```

When executing remote commands, **sshg3** exits with the status of the command run indicated with exit codes:

```
0     The remote command was run successfully.
127   The requested remote command was not found.
```

Examples

Connect as the local user name to host *`remotehost`*, port 2222, and open shell:

```
$ sshg3 remotehost#2222
```

Connect to the host specified by the connection profile *`profile1`* in the `ssh-broker-config.xml` file, and run the `who` command (and exit after running the command):

```
$ sshg3 profile1 who
```

Connect as *`user`* to host *`remotehost`*, and open a local port forwarding from port 143 on the client to port 143 on *`imapserver`*. Do not open shell. Also other hosts may connect to the local port. The connection from *`remotehost`* to *`imapserver`* will not be secured:

```
$ sshg3 -L 143:imapserver:143 -g -S user@remotehost
```

scpg3

scpg3 — Secure Shell file copy client - Generation 3

Synopsis

`scpg3` [`options` ...]
[*`src_profile`*: | [*`user@`*] *`src_host`* [*`#port`*]:] *`src_file`* ...
[*`dst_profile`*: | [*`user@`*] *`dst_host`* [*`#port`*]:] *`dst_file_or_dir`*

Description

scpg3 (**scpg3.exe** on Windows) is used to securely copy files over the network. **scpg3** launches **ssh-broker-g3** to provide a secure transport using the Secure Shell version 2 protocol. **ssh-broker-g3** will ask for passwords or passphrases if they are needed for authentication. **scpg3** uses the configuration specified in the `ssh-broker-config.xml` file.

Copies between two remote hosts are permitted. The remote host(s) must be running a Secure Shell version 2 server with the **sftp-server** (or **sft-server-g3**) subsystem enabled. Tectia Server has **sft-server-g3** enabled by default.

Any filename may contain a host, user, and port specification to indicate that the file is to be copied to or from that host ([*`user@`*] *`host`* [*`#port`*]). If no user name is given, the local user name is assumed. If no port is given, the default Secure Shell port 22 is assumed. Alternatively, a connection profile defined in the `ssh-broker-config.xml` file (*`profile`*) can be given.

Note

When entering a connection profile in the **scpg3** command, note that Tectia Client deduces the meaning of the argument differently depending on its format. If there is an `@` sign in the given attribute value, Tectia Client always interprets it to be `<username@hostname>`, i.e. not a profile.

Also, if there are dots in a profile name (for example `host.x.example.com`, the dots need to be escaped on command line. On Unix, enter `host\.x\.example\.com`, instead. On Windows, enter `host~.x~.example~.com`, instead. Otherwise the profile name is taken as a host name and the current Windows user name is used for logging in.

The *`host`* parameter can optionally be enclosed in square brackets ([]) to allow the use of semicolons. The *`file`* argument can contain simple wildcards: asterisk (*****) for any number of any characters, and question mark (**?**) for any one character.

For information on special characters in file names, see the section called “Filename Support”.

Options

The following command-line parameters can be used to further specify the **scpg3** options.

`-4`

Defines that all connection-related DNS resolutions will be resolved as an IPv4 address.

`-6`

Defines that all connection-related DNS resolutions will be resolved as an IPv6 address.

`-a` [*`arg`*]

Transfers files using the ASCII mode, that is, newlines will be converted on the fly. For transfers between Tectia on z/OS and other hosts, this also enables automatic ASCII-EBCDIC conversions. See the **sftpg3 ascii** command in the section called “Commands”.

If the server does not advertise the newline convention, and you are not using a host profile that specifies its host type, you can give **scpg3** a hint by giving an argument after `-a`. The default is to set the destination newline convention, but you can specify either one by prefixing the argument with `src:` or `dest:` for source or destination convention, respectively. The available conventions are `dos`, `unix`, and `mac`, using `\r\n`, `\n`, and `\r` as newlines, respectively. Note that there is no space between the `-a` and its argument. An example is shown below:

```
$ scpg3 -asrc:unix -adest:dos src_host:src_file dest_host:dest_file
```

To force the newline conventions, use these values: `force-dos`, `force-unix`, and `force-mac`. These settings force the newline mode irrespective of what the remote SSH server suggests to the SCP client.

`-B, --batch-mode`

Uses batch mode. Fails authentication if it requires user interaction on the terminal.

Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

`-b` *`buffer_size_bytes`*

Defines the maximum buffer size for one SFTP protocol read or write request (default: *`32768`* bytes).

The maximum number of SFTP protocol read or write requests sent in parallel within the transfer of a single file can be specified with the `-N` option.

Note that when streaming (see `--streaming`) is used (as it is by default when transferring files larger than *`buffer_size_bytes`* to/from Tectia Server), this option is not used for defining buffer sizes.

`-C`

Disables compression from the current connection.

`+C`

Enables zlib compression for this particular connection.

`-c, --ciphers=` *`LIST`*

Sets the allowed ciphers in their preferred order offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers AEAD_AES_128_GCM,seed-cbc@ssh.com,aes256-cbc
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported cipher names.

`-D, --debug=` *`LEVEL`*

Sets the debug level. *`LEVEL`* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

Note

Option `-D` only applies on Unix. On Windows, instead of this command-line tool, use the Connection Broker debugging options `-D, -l`.

Note

The debug level can be set only when the **scpg3** command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

`-d`

Forces target to be a directory.

`-I, --interactive`

Prompts whether to overwrite an existing destination file (does not work with `-B`).

`-i` *`FILE`*

Defines that private keys defined in the identification file are used for public-key authentication.

`-K, --identity-key-file=` *`FILE`*

Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple `-K` options on command line.

`-m` *`fileperm`* [:*`dirperm`*]

This option can be used only on Windows. Sets the default file and directory permission bits for file upload to Unix servers.

`-N` *`max_requests`*

Defines the maximum number of SFTP protocol read or write requests sent in parallel (default: 10).

The size of the buffer used in each read or write request can be specified with the `-b` option.

Note that this value applies within the transfer of a single file; it cannot be used to define the number of files sent in parallel.

When streaming (see `--streaming`) is used (as it is by default when transferring files larger than *`buffer_size_bytes`* specified with the `-b` option to/from Tectia Server), this option is not used.

`-O, --offset=` `r` `<`*`offset`*`>` | `w` `<`*`offset`*`>` | `l` `<`*`length`*`>` | `t` `<`*`length`*`>`

Sets offset. Offset `r<`*`offset`*`>` specifies the start offset in the source file. Offset `w<`*`offset`*`>` specifies the start offset in the destination file. Length `l<`*`length`*`>` specifies the amount of data to be copied. Truncate length `t<`*`length`*`>`, if given, specifies the length to which the destination file is truncated or expanded after the file data has been copied.

`-p`

Preserves the file permissions and the timestamps when both the source and the destination are on Unix file systems (including z/OS USS). Preserves the timestamps but not the file permissions, if either one, the source or the destination is on Windows. If the destination is on z/OS MVS, none will be preserved.

`-P` *`port`*

Connects to this Secure Shell port on the remote machine (default: *`22`*).

`-Q`

Does not show progress indicator. The effect of this option is the same as using `--progress-display=no`.

Do not use this option together with parameter `--statistics`.

`-q`

Uses quiet mode (only fatal errors are shown). This option overrides the `quiet-mode` setting made in the Connection Broker configuration file.

`-r`

Recurses subdirectories.

`-u, --unlink-source`

Removes source files after copying (file move).

`-v, --verbose`

Uses verbose mode (equal to `-D 2`).

`-W, --whole-file`

Does not try incremental checks. By default (if this option is not given), incremental checks are tried. This option can only be used together with the `--checksum` option.

`--aa, --allowed-authentications=` *`METHODS`*

Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. For example:

```
--allowed-authentications keyboard-interactive,password
```

Enter `help` as the value to view the currently supported authentication methods.

`--any-alg`

Allow any supported cipher, mac, kex, hostkey and publickey algorithm to be used.

`--append`

Appends data to the end of the destination file.

`--binary`

Uses binary transfer mode. If the server is Tectia Server for IBM z/OS, the server is requested not to perform ASCII to EBCDIC conversion, and the file is transferred using the Stream format. You can use the `--src-site` and `--dst-site` options to change the values.

`--checkpoint=b <`*`bytes`*`>`

Byte interval between checkpoint updates (default: 10 MB). This option can only be used when `--checksum=checkpoint`.

`--checkpoint=s <`*`seconds`*`>`

Time interval between checkpoint updates (default: *`10`* seconds). This option can only be used when `--checksum=checkpoint`.

`--checksum [ =yes | no | md5 | sha1 | sha256 | sha512 | md5-force | sha1-force | sha256-force | sha512-force | checkpoint ]`

Uses MD5, SHA-1 or SHA-2 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than *`buffer_size_bytes`* are not checked unless forced, i.e. `sha1-force`. (default: `yes`, uses SHA-1 checksums in FIPS mode, MD5 checksums otherwise). Use checkpointing when transferring large files one by one.

`--compressions=` *`METHODS`*

Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter `help` as the value to view the currently supported compression methods.

`--dst-site=` *`PARAM`*

Uses the specified site parameters with the destination files. See the **sftpg3 site** command in the section called "Commands".

`--exclusive`

Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

`--fips`

Performs the checksums using the FIPS cryptographic library.

`--force-lower-case`

Destination filename will be converted to lowercase characters.

`--hostkey-algorithms=` *`HOSTKEYALGORITHMS`*

Sets the allowed host key algorithms to be offered to the server. List the host key algorithms in a comma-separated list. For example:

```
--hostkey-algorithms ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported host key algorithms.

`--overwrite` [`=yes` | `no`]

Selects whether to overwrite existing destination file(s) (default: `yes`).

`--identity=` *`ID`*

Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

`--identity-key-hash=` *`ID`*

Defines the private key used in user authentication with the corresponding public key hash.

`--identity-key-id=` *`ID`*

Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

`--keep-alive=` *`VALUE`*

Defines how often keep-alive messages (non-operation packages) are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

`--kexs=` *`kexs`*

Sets the allowed key exchange (KEX) methods to be offered to the server. List the KEX names in a comma-separated list. For example:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported KEX methods.

`--kip`

Defines keyboard-interactive and password as the allowed methods for user authentication; the same as

```
--allowed-authentications keyboard-interactive,password
```

`--macs=` *`LIST`*

Sets the allowed MACs in preferred order to be offered to the server. List the MAC names in comma-separated list. For example:

```
--macs hmac-sha2-256,hmac-sha1-96,hmac-md5
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported MAC names.

`--password=` *`PASSWORD`* | `file://` *`PASSWORDFILE`* | `extprog://` *`PROGRAM`*

Sets user password that the client will send as a response to password authentication. The *`PASSWORD`* can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (`--password=file://`*`PASSWORDFILE`*), or entering a path to a program or script that outputs the password (`--password=extprog://`*`PROGRAM`*).

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

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

Caution

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

`--plugin-path=` *`PATH`*

Sets plugin path to *`PATH`*. This is only used in the FIPS mode.

`--prefix=` *`PREFIX`*

Adds a prefix to a filename during the file transfer. The prefix is removed after the file has been successfully transferred.

On z/OS, when applied to MVS data set names, the prefix is inserted after the High Level Qualifier (HLQ). In case you want the prefix to be a separate qualifier, include a dot at the end of the prefix:

```
--prefix=PREFIX.
```

`--publickey-algorithms=` *PUBLICKEY_ALGORITHMS*

Allow only selected signature algorithms to be used in public key authentication. For example:

```
--publickey-algorithms=x509v3-ssh-rsa,rsa-sha2-512
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the available signature algorithms.

`--src-site=` *PARAM*

Uses the specified site parameters with the source files. See the **site** command in the section called "Commands".

`--statistics` [`=no` | `yes` | `simple`]

Note

In release 6.1.5, the behavior of the `--statistics` option has changed and the `--statistics-format` option has been removed. Instead of them, use the new `--summary-display` and `--summary-format` options.

The `--statistics` option chooses the style of the statistics to be shown after a file transfer operation. Note that `--statistics` and `--summary-display` must not be used together.

The `--statistics` option takes the following values:

`no` - no statistics will be created.

`yes` - shows a progress bar during the file transfer. This is the default. An example of the output:

```
scpg3 --statistics="yes" sourcefile destinationfile
sourcefile                |  127MB |  42.9MiB/s | TOC: 00:00:03 | 100%
```

`simple` - simple one-line statistics will be displayed after the file transfer has ended. For example:

```
scpg3 --statistics=simple sourcefile destinationfile
sourcefile   | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`--summary-display` [`=no` | `yes` | `simple` | `bytes`]

Chooses the style of the file transfer summary data to be displayed after a file transfer operation. With the summary display, the progress bar data is also displayed by default.

Note that `--summary-display` and `--statistics` must not be used together.

The `--summary-display` option takes the following values:

`no` - no summary data will be created. This is the default.

`yes` - detailed summary data will be created. You can configure the contents with the `summary-format` option. By default, the following contents are displayed in the summary:

```
Default settings:                    Render for example this:
"Source: %c:%g\n"                    user@host1#22:/path/to/source/file
"Source parameters: %e\n"            X=TEXT, C=ISO8859-1,D=IBM.1047
"Destination: %C:%G\n"               user@host2#22:/path/to/destination/file
"Destination parameters: %E\n"       NONE
"File size: %s bytes\n"              123456 bytes
"Transferred: %t bytes\n"            123456 bytes
"Rate: %RB/s\n"                      345kiB/s
"Start: %xy-%xt-%xd %xh:%xm:%xs\n"   2010-01-26 13:10:56
"Stop: %Xy-%Xt-%Xd %Xh:%Xm:%Xs\n"    2010-01-26 13:23:30
"Time: %y\n"                         00:12:34
```

`simple` - simple one-line summary will be displayed. For example:

```
scpg3 --summary-display=simple sourcefile destinationfile
sourcefile | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`bytes` - basic statistics reporting the transferred bytes will be displayed. For example:

```
scpg3 --summary-display=bytes sourcefile destinationfile
Transferred 12915145984 bytes, file: 'sourcefile' -> 'destinationfile'
```

`--summary-format=` *`FORMAT_STRING`*

Chooses the format and the contents of the summary. You can use this option when `--summary-display=yes`. Do not use this option with `--statistics`.

Select the contents for the summary using the following definitions:

```
%c  - source connection: user@host#port or profile
%C  - destination connection: user@host#port or profile
%D* - current date
%e  - source parameters (file transfer and data set parameters)
%E  - destination parameters (file transfer and data set parameters)
%f  - source file name
%F  - destination file name
%g  - /path/to/source/file
%G  - /path/to/destination/file
%k  - compression done ("zlib" or "none")
%p  - transfer percentage
%q  - transfer rate in bit/s
%Q  - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r  - transfer rate in bytes/s
%R  - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%s  - file size in bytes
%S  - file size as "XXyB" (B, kiB, MiB or GiB)
%t  - transfer size in bytes
%T  - transfer size as "XXyB" (B, kiB, MiB or GiB)
%x* - start date
%X* - end date
%y  - elapsed time
%Y  - time remaining
```

```
%z  - ETA or TOC, if transfer has finished
%Z  - string "ETA" or "TOC", if transfer has finished

Where * is one of the following:

h - hours (00-23)
m - minutes (00-59)
s - seconds (00-59)
f - milliseconds (0-999)
d - day of the month (1-31)
t - month (1-12)
y - year (1970-)

Other special characters in format strings are:

\n - line feed
\r - carriage return
\t - horizontal tab
\\ - backslash
```

`--progress-display` [`=no` | `bar` | `line`]

Chooses the mode of displaying the progress during a file transfer operation. The default is `bar`, which shows a progress bar. Option `line` shows the progress information according to the settings made in the `--progress-line-format` option.

Do not use this option with `--statistics`.

`--progress-line-format=` *`FORMAT_STRING`*

Chooses what information will be shown on the progress line. You can use this option when `--progress-display=line`.

Do not use this option with `--statistics`.

Select the contents for the progress line using the definitions described for command: `--summary-format`

`--progress-line-interval=` *`seconds`*

Defines how often the progress information is updated in line mode. The interval is given in seconds, and the default is 60 seconds.

Do not use this option with `--statistics`.

`--streaming` [`=yes` | `no` | `force` | `ext`]

Uses streaming in file transfer, if server supports it. Files smaller than *`buffer_size_bytes`* are not transferred using streaming. Use `force` with small files. Default: `yes`

Use `ext` with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The `--streaming=ext` option requires also the `--checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

`--sunique`

Stores files with unique names. In case more than one of the transferred files have the same name, this feature adds a sequential number to the end of the repeated file name, for example: `file.name`, `file.name1`, and `file.name2`.

`--tcp-connect-timeout=` *`VALUE`*

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter the timeout value as a positive number. Value 0 (zero) disables this feature and the default system TCP timeout will be used instead.

`--template-profile` *`profile`*

Use the specified profile when connecting.

`--user=` *`USERNAME`*

Logs in using this user name if the user name is not provided in the address string.

`-V, --version`

Displays program version and exits.

`-h, --help, -?`

Displays a short summary of command-line options and exits.

Filename Support

Different operating systems allow different character sets in filenames. On Unix, some of the special characters are allowed in filenames, but on Windows, the following characters are not allowed:

```
\/ : * ? " < > |
```

When you use the **scpg3** command to copy files with special characters (for example `unixfilename*?".txt`) from a Unix server to Windows, you need to provide the files with new names that are acceptable on Windows. Enter the commands in the following format:

```
$ scpg3 user@unixserver:"unixfilename~*~?\".txt" windowsfilename.txt
```

The general rule is to follow your platform specific syntax when you enter filenames containing special characters as arguments to the **scpg3** command.

Tectia fully supports filenames containing only ASCII and UTF-8 characters. Filenames containing characters from other character sets are not guaranteed to work.

Using Wildcards

The **scpg3** command supports `*` and `?` as wildcards.

The wildcards can be used both on the remote and the local side in the commands. The following example command will copy all text files (`*.txt`) from all subdirectories of directory `dir2` whose names begin with the prefix `data-` into the current local directory (`.`):

```
$ scpg3 -r user@server:"dir2/data-*/*.txt" .
```

Note that on Unix, the characters ***** and **?** can appear also in the filenames. So it is necessary to use escape characters to distinguish the wildcards from the characters belonging to a filename. See more information in the section called “Escaping Special Characters”.

Escaping Special Characters

Some special characters that are used in filenames in different operating system, may have a special meaning in the Tectia commands. Note also that the meaning can be different in various parts of the file transfer system.

In the **scpg3** command, the following characters have a special meaning, and they need to be escaped in commands that take filenames as arguments:

***** asterisk is a wildcard for any number of any characters

? question mark is a wildcard for any single character

"" quotation marks placed around strings that are to be taken 'as is'

**** backslash is an escape character on Unix

~ tilde is an escape character on Windows.

The escape character tells the **scpg3** command to treat the next character "as is" and not to assume any special meaning for it. The escape character is selected according to the operating system of the local machine.

Note that the **** and **~** characters are special characters themselves, and if they are present in the filename, escape characters must be placed in front of them, too. Therefore, if you need to enter a filename containing **** in Unix or **~** in Windows to the **scpg3** command, add the relevant escape character to it:

**** on Unix

~~ on Windows

See the examples below to learn how the escape characters are used in the Tectia **scpg3** command, and how to enter filenames with special characters in different operating systems.

Examples of filenames in the **scpg3** command:

The following filenames are valid in Unix, but they need escape characters in the commands:

```
file|name.txt
```

```
file-"name".txt
file?name.txt
file*name.txt
file\name.txt
file - name.txt
file~name.txt
```

When using the **scpg3** command on Unix, in certain cases several escape characters are needed, as they escape one another. Enter the above mentioned filenames in the following formats:

```
file\|name.txt     or  "file|name.txt"
file-\"name\".txt
file\\\?name.txt   or  "file\?name.txt"
file\\\*name.txt   or  "file\*name.txt"
file\\\\name.txt   or  "file\\\name.txt"
file\ -\ name.txt  or  "file - name.txt"
file~name.txt
```

Example commands on Unix:

```
$ scpg3 user@server:file\\\*name.txt .
```

```
$ scpg3 user@server:file\ -\ name.txt .
```

When using the **scpg3** command on Windows, enter the above mentioned Unix filenames in the following formats:

```
"file|name.txt"
file-\"name\".txt     (Note that Windows requires \ to escape the " character)
"file~?name.txt"
"file~*name.txt"
file~\name.txt
"file - name.txt"
file~~name.txt
```

The operating system interprets the quotation marks ("") here so that the **scpg3** command receives the string without the quotation marks as a parameter.

Example commands on Windows:

```
> scpg3 user@server:"file~*name.txt" filename.txt
```

```
> scpg3 user@server:"file - name.txt" .
```

Environment Variables

scpg3 uses the following environment variables:

SSH_SFTP_CHECKSUM_MODE `=yes|no|md5|sha1|sha256|sha512|md5-force|sha1-force|sha256-force|sha512-force|checkpoint`

Defines the setting for comparing checksums. For more information on the available values, see **checksum** .

SSH_SFTP_SHOW_BYTE_COUNT `=yes|no`

If this variable is set to `yes`, the number of transferred bytes is shown after successful file transfer. Also the names of source and destination files are shown. The default is `no`.

SSH_SFTP_STATISTICS `=yes|no|simple`

If this variable is set to `yes` (default), normal progress bar is shown while transferring the file. If it is set to `no`, progress bar is not shown. If it is set to `simple` file transfer statistics are shown after the file has been transferred.

UTF8_MODE `=0|1`

If this variable is set to `0`, the default charset mode UTF-8 is disabled. To force UTF-8 charset mode for file names, set the variable to `1`.

Exit Values

scpg3 returns the following values based on the result of the operation:

```
0     Operation was successful.
1     Internal error.
2     Connection aborted by the user.
3     Destination is not a directory, but a directory was specified by the user.
4     Connecting to the host failed.
5     Connection lost.
6     File does not exist.
7     No permission to access file.
8     Undetermined error from sshfilexfer.
11    Some non-fatal errors occured during a directory operation.
101   Wrong command-line arguments specified by the user.
```

Examples

Copy files from your local system to a remote Unix system:

```
$ scpg3 localfile user@remotehost:/dst/dir/
```

Copy files from your local system to a remote Windows system:

```
$ scpg3 localfile user@remotehost:/C:/dst/dir/
```

Copy files from a remote system to your local disk:

```
$ scpg3 user@remotehost:/src/dir/srcfile /dst/dir/dstfile
```

Copy files from one remote system to another using connection profiles defined in the `ssh-broker-config.xml` file:

```
$ scpg3 profile1:/src/dir/srcfile profile2:/dst/dir/dstfile
```

sftpg3

sftpg3 — Secure Shell file transfer client - Generation 3

Synopsis

`sftpg3` [`options` ...]
[*`profile`* | [*`user@`*] *`host`* [*`#port`*] [*`:path`*] | sftp:// [[*`user`*] [*`:password`*] @] *`host`* [*`:port`*] [*`/path`*]]

Description

sftpg3 (**sftpg3.exe** on Windows) is an FTP-like client that can be used for secure file transfer over the network. **sftpg3** launches **ssh-broker-g3** to provide a secure transport using the Secure Shell version 2 protocol. **ssh-broker-g3** will ask for passwords or passphrases if they are needed for authentication. **sftpg3** uses the configuration specified in the `ssh-broker-config.xml` file.

When started interactively, **sftpg3** displays a prompt where the SFTP commands can be entered. It is also possible to start **sftpg3** non-interactively with a batch file that contains the commands to be run. For information on the available commands, see the section called “Commands”.

As an alternative to using the command line to set default values for various parameters, it is possible to define the commands in a startup batch file that is run each time **sftpg3** is started. By default, **sftpg3** looks for a file named `ssh_sftp_batch_file` in the user-specific directory `$HOME/.ssh2/` on Unix or `%APPDATA%\SSH\` on Windows.

sftpg3 has two connection end points, local and remote, and both of them can be connected to other hosts than the SFTP client host. If started without arguments, the local end point is connected to the file system of the SFTP client host and the remote end point is unconnected. The connected host(s), with the exception of the SFTP client host, must be running a Secure Shell version 2 server with the **sftp-server** (or **sft-server-g3**) subsystem enabled. Tectia Server has **sft-server-g3** enabled by default.

The remote connection end point can be given directly as an argument to the **sftpg3** command or it can be given with the **open** SFTP command after **sftpg3** has started. The local connection end point can be given with the **lopen** SFTP command.

When connecting, you can give either the name of a connection profile defined in the `ssh-broker-config.xml` file (*`profile`*) or the IP address or DNS name of the remote host, optionally with the remote user name and the port of the Secure Shell server ([*`user@`*] *`host`* [*`#port`*] [*`:path`*]). If no user name is given, the local user name is assumed. If no port is given, the default Secure Shell port 22 is assumed. The path can be used to specify the initial working directory the sftpg3 will use. The host, port, username etc. can also be given with sftp URL syntax. Using this format user password can also be provided.

Note

When entering a connection profile in **sftpg3**, note that Tectia Client deduces the meaning of the argument differently depending on its format. If there is an @ sign in the given attribute value, Tectia Client always interprets it to be *<username@hostname>*, i.e. not a profile.

Also, if there are dots in a profile name (for example `host.x.example.com`, the dots need to be escaped on command line. On Unix, enter `host\.x\.example\.com`, instead. On Windows, enter `host~.x~.example~.com`, instead. Otherwise the profile name is taken as a host name and the current local user name is used for logging in.

For information on special characters in filenames, see the section called “Filename Support”.

Options

The following options are available:

`-4`

Defines that all connection-related DNS resolutions will be resolved as an IPv4 address.

`-6`

Defines that all connection-related DNS resolutions will be resolved as an IPv6 address.

`-b` *`buffer_size_bytes`*

Defines the maximum buffer size for one SFTP protocol read or write request (default: *`32768`* bytes).

The maximum number of SFTP protocol read or write requests sent in parallel within the transfer of a single file can be specified with the `-N` option.

Note that when streaming (see `--streaming`) is used (as it is by default when transferring files larger than *`buffer_size_bytes`* to/from Tectia Server), this option is not used for defining buffer sizes.

`-B` [`-` | *`batch_file`*]

The `-B -` option enables reading from the standard input. This option is useful when you want to launch processes with **sftpg3** and redirect the stdin pipes.

By defining the name of a *`batch_file`* as an attribute, you can execute SFTP commands from the given file in batch mode. The file can contain any allowed SFTP commands. For a description of the commands, see the section called “Commands”.

Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

`-C`

Disables compression from the current connection.

`+C`

Enables zlib compression for this particular connection.

`-c, --ciphers=` *`LIST`*

Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported cipher names.

`-D, --debug=` *`LEVEL`*

Sets the debug level. *`LEVEL`* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

Note

Option `-D` only applies on Unix. On Windows, instead of this command-line tool, use the Connection Broker debugging options `-D, -l`.

Note

The debug level can be set only when the **sftpg3** command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

`-i` *`FILE`*

Defines that private keys defined in the identification file are used for public-key authentication.

`-K, --identity-key-file=` *`FILE`*

Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple `-K` options on command line.

`-N` *`max_requests`*

Defines the maximum number of SFTP protocol read or write requests sent in parallel (default: 10).

The size of the buffer used in each read or write request can be specified with the `-b` option.

Note that this value applies within the transfer of a single file; it cannot be used to define the number of files sent in parallel.

When streaming (see `--streaming`) is used (as it is by default when transferring files larger than *`buffer_size_bytes`* specified with the `-b` option to/from Tectia Server), this option is not used.

`-P` *`port`*

Connects to this Secure Shell port on the remote machine (default: *`22`*).

`-q, --quiet`

Suppresses the printing of error, warning, and informational messages. This option overrides the `quiet-mode` setting made in the Connection Broker configuration file.

`-v, --verbose`

Uses verbose mode (equal to `-D 2`).

`--aa, --allowed-authentications=` *`METHODS`*

Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. For example:

```
--allowed-authentications keyboard-interactive,password
```

Enter `help` as the value to view the currently supported authentication methods.

`--any-alg`

Allow any supported cipher, mac, kex, hostkey and publickey algorithm to be used.

`--compressions=` *`METHODS`*

Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter `help` as the value to view the currently supported compression methods.

`--exclusive`

Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

`--fips`

Performs the checksums using the FIPS cryptographic library.

`--hostkey-algorithms=` *`algorithms`*

Sets the allowed host key algorithms to be offered to the server. List the host key algorithms in a comma-separated list. For example:

```
--hostkey-algorithms ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported host key algorithms.

`--identity=` *`ID`*

Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

`--identity-key-hash=` *`ID`*

Defines the private key used in user authentication with the corresponding public key hash.

`--identity-key-id=` *`ID`*

Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

`--keep-alive=` *`VALUE`*

Defines how often keep-alive messages are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

`--kexs=` *`kexs`*

Sets the allowed key exchange (KEX) methods to be offered to the server. List the KEX names in a comma-separated list. For example:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported KEX methods.

`--kip`

Defines keyboard-interactive and password as the allowed methods for user authentication; the same as

```
--allowed-authentications keyboard-interactive,password
```

`--macs=` *`LIST`*

Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

```
--macs hmac-sha1-96,hmac-md5,hmac-md5-96
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the currently supported MAC names.

`--password=` *`PASSWORD`* | `file://` *`PASSWORDFILE`* | `extprog://` *`PROGRAM`*

Sets the user password or passphrase that the client will send as a response to an authentication method requesting a password or passphrase (hereafter: password). This can be used also with password-protected certificates and public-keys.

The *PASSWORD* can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (`--password=file://`*PASSWORDFILE*), or entering a path to a program or script that outputs the password (`--password=extprog://`*PROGRAM*).

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

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

Caution

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

`--plugin-path=` *PATH*

Sets plugin path to *PATH*. This is only used in the FIPS mode.

`--publickey-algorithms=` *PUBLICKEY_ALGORITHMS*

Allow only selected signature algorithms to be used in public key authentication. For example:

```
--publickey-algorithms=x509v3-ssh-rsa,rsa-sha2-512
```

Value `any` allows all supported algorithms. Enter `help` as the value to view the available signature algorithms.

`--tcp-connect-timeout=` *VALUE*

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter a timeout value as a positive number. Value 0 (zero) disables this feature and the default system TCP timeout will be used instead.

`--template-profile` *profile*

Use the specified profile when connecting.

`-l, --user=` *USERNAME*

USERNAME will be used in the logon if the user name is not specified in the address string.

`-V, --version`

Displays program version and exits.

`-h, --help, -?`

Displays a short summary of command-line options and exits.

Commands

When **sftpg3** is ready to accept commands, it will display the prompt `sftp>`. The user can then enter any of the following commands:

`!`	`append`	`ascii`	`auto`	`binary`
`break`	`cd`	`charset`	`chmod`	`close`
`continue`	`debug`	`delete`	`digest`	`echo`
`exit`	`get`	`getext`	`help`	`helpall`
`lappend`	`lcd`	`lcharset`	`lchmod`	`lclose`
`ldelete`	`ldigest`	`lls`	`llsroots`	`lmkdir`
`localopen`	`locsite`	`lopen`	`lpwd`	`lreadlink`
`lrename`	`lrm`	`lrmdir`	`ls`	`lsite`
`lsroots`	`lsymlink`	`mget`	`mkdir`	`mput`
`open`	`pause`	`put`	`pwd`	`quit`
`readlink`	`rename`	`rm`	`rmdir`	`set`
`setext`	`setperm`	`sget`	`site`	`sput`
`sunique`	`symlink`	`type`	`verbose`	

`!` [*`command`*] [*`arguments`*]

Invokes an interactive shell on the local machine. if a *`command`* is given, it is used as the command to be executed. Optional *`arguments`* can be given depending on the command.

`append` [`-u, --unlink-source`] [`--streaming`] [`--force-lower-case`] [`--statistics`] [`--summary-display`] [`--summary-format`] [`--progress-display`] [`--progress-line-format`] [`--progress-line-interval`] *`srcfile`* [*`dstfile`*]

Appends the specified local file to the remote file. No globbing can be used.

Options:

`-u, --unlink-source`

Removes the source file after file transfer.

`--streaming` [`=yes` | `no` | `force` | `ext`]

Uses streaming in file transfer if the server supports it. Files smaller than *`buffer_size_bytes`* are not transferred using streaming. Use `force` with small files. Default: `yes`

Use `ext` with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The `--streaming=ext` option requires also the `--checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

`--force-lower-case`

Destination filename will be converted to lowercase characters.

The semantics of options `--statistics`, `--summary-display`, `--summary-format`, `--progress-display`, `--progress-line-format`, and `--progress-line-interval` are the same as with **get**.

`ascii` [`-s`] [*`remote_nl_conv`*] [*`local_nl_conv`*]

Command **ascii** sets the transfer mode to ASCII.

For transfers between Tectia on z/OS and other hosts, this also enables automatic ASCII-EBCDIC conversion. Default conversion is between code sets ISO8859-1 and IBM-1047. Files are transferred using the `LINE` format. The **site** and **lsite** commands can be used to change the values.

If you enter the **ascii** command with any options, it does not set the transfer mode to ASCII, but affects the newline conventions used in the transferred files. You can also set the server's newline convention by using a host profile that specifies the host type. For more information, see the `host-type` attribute in the section called “The `profiles` Element”.

Options:

`-s`

Shows the current newline convention. The line delimiters used in different systems are:

```
dos:    CRLF (\r\n, 0x0d 0x0a)
mac:    CR (\r, 0x0d)
mvs:    NEL (\n, 0x15)
unix:   LF (\n, 0x0a)
```

`remote_nl_conv local_nl_conv`

This syntax can be used to define the remote and local newline conventions. The *`local_nl_conv`* option operates on the local end, but notice that usually the correct local newline convention is already compiled in.

You can either set hints of the newline conventions for the underlying transfer layer, which by default tries to use the actual newline convention given by the server, or alternatively you can force the newline mode.

To set hints of the newline conventions, use these values in the *`remote_nl_conv`* and *`local_nl_conv`* options: `dos`, `unix`, and `mac`. These settings will be used if the remote SSH server does not automatically provide any newline information to the SFTP client. For example:

```
sftp> ascii
 File transfer mode is now ascii.
sftp> ascii unix dos
```

```
Newline conventions updated.
```

To force the newline conventions, use these values: `force-dos`, `force-unix`, and `force-mac`. These settings force the newline mode irrespective of what the remote SSH server suggests to the SFTP client.

```
sftp> ascii
 File transfer mode is now ascii.
sftp> ascii force-unix force-dos
 Newline conventions updated.
```

You can also set either one of the options to `ask`, which will cause **sftpg3** to prompt you for the newline convention when needed.

`auto`

File transfer mode will be selected automatically from the file extension.

`binary`

Files will be transferred in binary mode.

`break`

Interrupts batch file execution. Batch file execution can be continued with the **continue** command.

`bye`

Quits the application.

`cd` *`directory`*

Changes the current remote working directory.

`charset` [`-e` *`esc`*] [`-s`] *`CHARSET`*

,

`charset` [`--list`]

Sets the file name charset client expects from the server and lists available character sets. The default character set is UTF-8. Value 'none' will unset character set settings.

Options:

`-e esc`

Set the escape policy for characters that cannot be represented in the given charset. Options include 'fail', 'escape', 'escape-u', 'escape-uplus' and 'escape-xml'. Last is the default value.

`-s`

With this option, the client will send the charset name to the server in filename-charser@ssh.com extension to request for example UTF-8 to be used in case the connected server supports it.

`--list`

List available file name character sets.

`chmod [ -R ] [ -f ] [ -v ] OCTAL-MODE [ file ...]`

,

`chmod [ -R ] [ -f ] [ -v ] [ ugoa ] [ +-= ] [ rwxs ] [ file ...]`

With Unix files, sets file permissions of the specified file or files to the bit pattern *OCTAL-MODE* or changes the file permissions according to the symbolic mode `[ugoa][+-=][rwxs]`.

Options:

`-R`

Recursively changes files and directories.

`-f`

Uses silent mode (error messages are suppressed).

`-v`

Uses verbose mode (lists every file processed).

`close`

Closes the remote connection.

`continue`

Continues interrupted batch file execution.

`debug [ disable | no | debuglevel ]`

Disables or enables debug. With `disable` or `no`, debugging is disabled. Otherwise, sets *debuglevel* as debug level string, as per command-line option `-D`.

`delete [ -H, --hash ] [ -o, --offset ] [ -l, --length ] file`

Tries to delete a file or directory specified in *file*. The options are the same as for **rm**.

`digest [ -H, --hash ] [ -o, --offset ] [ -l, --length ] file`

Calculates MD5, SHA-1 or SHA-2 digest over file data. The digest is calculated over the data on the disk. If any code or line delimiter conversion attributes are in effect, they are ignored when calculating the digest.

Options:

`-H, --hash= [ alg ]`

Specify the hash algorithm. Support depends on server, values `md5`, `sha1`, `sha256` and `sha512` are commonly supported. (default: `sha1`).

`-o, --offset=` *`OFFSET`*

Start reading from file offset *`OFFSET`*.

`-l, --length=` *`LENGTH`*

Read *`LENGTH`* bytes of file data.

`echo` *`Text to be echoed.`*

Echo the text. This command can be used for example in batch mode to print text into batch logs.

`exit`

Quits the application.

`get [-p, --preserve-attributes] [-u, --unlink-source] [-I, --interactive] [--overwrite] [--checksum] [-W, --whole-file] [--checkpoint] [--streaming] [--force-lower-case] [--prefix] [--statistics] [--summary-display] [--summary-format] [--progress-display] [--progress-line-format] [--progress-line-interval] [--max-depth=]` *`file`* ...

Transfers the specified files from the remote end to the local end. By default, directories are recursively copied with their contents, but this is configurable in the Connection Broker with the SFTP compatibility mode setting (`sftpg3-mode` in `ssh-broker-config.xml` configuration), with the environment variable `SSH_SFTP_CMD_GETPUT_MODE` or the SFTP command **set compatibility-mode**=*`mode`*). To view the currently set SFTP compatibility mode, run command:

```
sftp> set
```

The currently set compatibility mode is also shown in the beginning of the help for command **help get**.

The SFTP compatibility mode options are:

`tectia`

The **sftpg3** client transfers files recursively from the current directory and all its subdirectories.

`ftp`

The **get** command is executed as **sget** meaning that it transfers a single file, and no subdirectories are copied.

`openssh`

Only regular files and symbolic links from the specified directory are copied, and no subdirectories are copied. Otherwise the semantics of the **get** command are unchanged.

Options:

`-p, --preserve-attributes`

Preserves the file permissions and the timestamps when both the source and the destination are on Unix file systems (including z/OS USS). Preserves the timestamps but not the file permissions,

if either one, the source or the destination is on Windows. If the destination is on z/OS MVS, none will be preserved.

`-u, --unlink-source`

Removes the source file after file transfer. Also directories are removed, if they become empty (move mode).

`-I, --interactive`

Prompts whether to overwrite an existing destination file (does not work with batch mode).

`--overwrite` [`=yes` | `no`]

Decides whether to overwrite existing destination file(s) (default: `yes`).

`--checksum` [`=yes` | `no` | `md5` | `sha1` | `sha256` | `sha512` | `md5-force` | `sha1-force` | `sha256-force` | `sha512-force` | `checkpoint`]

Uses MD5, SHA-1 or SHA-2 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than *`buffer_size_bytes`* are not checked unless forced, i.e. `sha1-force`. (default: `yes`, uses SHA-1 checksums in FIPS mode, MD5 checksums otherwise). Use checkpointing when transferring large files one by one.

`-W, --whole-file`

Does not try incremental checks. By default (if this option is not given), incremental checks are tried. This option can only be used together with the `--checksum` option.

`--checkpoint=s` <*`seconds`*>

Time interval between checkpoint updates (default: *`10`* seconds). This option can only be used when `--checksum=checkpoint`.

`--checkpoint=b` <*`bytes`*>

Byte interval between checkpoint updates (default: 10 MB). This option can only be used when `--checksum=checkpoint`.

`--streaming` [`=yes` | `no` | `force` | `ext`]

Uses streaming in file transfer if the server supports it. Files smaller than *`buffer_size_bytes`* are not transferred using streaming. Use `force` with small files. Default: `yes`

Use `ext` with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The `--streaming=ext` option requires also the `--checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

An alternative way to activate extended streaming is to define `SSH_SFTP_STREAMING_MODE=ext` and `SSH_SFTP_CHECKSUM_MODE=no` as environment variables.

`--force-lower-case`

Destination file name will be converted to lower case characters.

`--max-depth=` *`VALUE`*

Defines whether directories are copied recursively. The value can be:

`0` - unlimited recursion, directories are recursively copied with their contents

`1` - copies files from the specified directory only, not from subdirectories

`2-n` - copies files recursively from the specified number of directory levels. Here `n` means the system-specific maximum.

This command line option overrides the recursion depth set in the Connection Broker configuration with element `sftpg3-mode` and/or the setting made using environment variable `SSH_SFTP_CMD_GETPUT_MODE`.

`--prefix=` *`PREFIX`*

Adds prefix `PREFIX` to filename during the file transfer. The prefix is removed after the file has been successfully transferred.

On z/OS, when applied to MVS data set names, the prefix will be inserted after the High Level Qualifier (HLQ) by default. In case you want the prefix to be a separate qualifier, include a dot at the end of the prefix:

```
--prefix=PREFIX.
```

`--statistics [ =no | yes | simple ]`

Note

In release 6.1.5, the behavior of the `--statistics` option has changed and the `--statistics-format` option has been removed. Instead of them, use the new `--summary-display` and `--summary-format` options.

The `--statistics` option chooses the style of the statistics to be shown after a file transfer operation. Note that `--statistics` and `--summary-display` must not be used together.

The `--statistics` option takes the following values:

`no` - no statistics will be created.

`yes` - shows a progress bar during the file transfer. This is the default. An example of the output:

```
sftp> get --statistics="yes" sourcefile
sourcefile                |  127MB |  42.9MiB/s | TOC: 00:00:03 | 100%
```

`simple` - simple one-line statistics will be displayed after the file transfer has ended. For example:

```
sftp> get --statistics=simple testfile
sourcefile   | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`--summary-display` [`=no` | `yes` | `simple` | `bytes`]

Chooses the style of the file transfer summary data to be displayed after a file transfer operation. With the summary display, the progress bar data is also displayed by default.

Note that `--summary-display` and `--statistics` must not be used together.

The `--summary-display` option takes the following values:

`no` - no summary data will be created. This is the default.

`yes` - detailed summary data will be created. You can configure the contents with the `summary-format` option. By default, the following contents are displayed in the summary:

```
Default settings:                    Render for example this:
"Source: %c:%g\n"                    user@host1#22:/path/to/source/file
"Source parameters: %e\n"            X=TEXT, C=ISO8859-1,D=IBM.1047
"Destination: %C:%G\n"               user@host2#22:/path/to/destination/file
"Destination parameters: %E\n"       NONE
"File size: %s bytes\n"              123456 bytes
"Transferred: %t bytes\n"            123456 bytes
"Rate: %RB/s\n"                      345kiB/s
"Start: %xy-%xt-%xd %xh:%xm:%xs\n"   2010-01-26 13:10:56
"Stop: %Xy-%Xt-%Xd %Xh:%Xm:%Xs\n"    2010-01-26 13:23:30
"Time: %y\n"                         00:12:34
```

`simple` - simple one-line summary will be displayed. For example:

```
sftp> get --summary-display=simple sourcefile
sourcefile  | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

`bytes` - basic statistics reporting the transferred bytes will be displayed. For example:

```
sftp> get --summary-display=bytes sourcefile
Transferred 12915145984 bytes, file: 'sourcefile' -> 'destinationfile'
```

`--summary-format=` *`FORMAT_STRING`*

Chooses the format and the contents of the summary. You can use this option when `--summary-display=yes`. Do not use this option with `--statistics`.

Select the contents for the summary using the following definitions:

```
%c  - source connection: user@host#port or profile
%C  - destination connection: user@host#port or profile
%D* - current date
%e  - source parameters (file transfer and data set parameters)
%E  - destination parameters (file transfer and data set parameters)
%f  - source file name
%F  - destination file name
```

```
%g  - /path/to/source/file
%G  - /path/to/destination/file
%k  - compression done ("zlib" or "none")
%p  - transfer percentage
%q  - transfer rate in bit/s
%Q  - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r  - transfer rate in bytes/s
%R  - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%s  - file size in bytes
%S  - file size as "XXyB" (B, kiB, MiB or GiB)
%t  - transfer size in bytes
%T  - transfer size as "XXyB" (B, kiB, MiB or GiB)
%x* - start date
%X* - end date
%y  - elapsed time
%Y  - time remaining
%z  - ETA or TOC, if transfer has finished
%Z  - string "ETA" or "TOC", if transfer has finished

Where * is one of the following:

h - hours (00-23)
m - minutes (00-59)
s - seconds (00-59)
f - milliseconds (0-999)
d - day of the month (1-31)
t - month (1-12)
y - year (1970-)

Other special characters in format strings are:

\n - line feed
\r - carriage return
\t - horizontal tab
\\ - backslash
```

`--progress-display` [`=no` | `bar` | `line`]

Chooses the mode of displaying the progress during a file transfer operation. The default is `bar`, which shows a progress bar. Option `line` shows the progress information according to the settings made in the `--progress-line-format` option.

Do not use this option with `--statistics`.

`--progress-line-format=` *`FORMAT_STRING`*

Chooses what information will be shown on the progress line. You can use this option when `--progress-display=line`.

Do not use this option with `--statistics`.

Select the contents for the progress line using the definitions described for option `--summary-format` of the **get** command above.

`--progress-line-interval=` *`seconds`*

Defines how often the progress information is updated in the line mode. The interval is given in seconds, and the default is 60 seconds.

Do not use this option with `--statistics`.

`getext`

Displays the extensions that will be ASCII in the auto transfer mode.

`lappend [ options ...] srcfile [ dstfile ]`

The same as **append**, but appends the specified remote file to the local file.

`lcd directory`

Changes the current local working directory.

`lcharset [ -e fail|escape|escape-u|escape-uplus|escape-xml ] CHARSET`

,

`lcharset [ --list ]`

The same as **charset**, but sets expected charset for local files.

`lchmod [ -R ] [ -f ] [ -v ] OCTAL-MODE [ file ...]`

,

`lchmod [ -R ] [ -f ] [ -v ] [ ugoa ] [ +-= ] [ rwxs ] [ file ...]`

The same as **chmod**, but operates on local files.

`lclose`

Closes the local connection.

`ldelete [ options ...] file ...`

The same as **delete**, but operates on local files.

`ldigest [ -H, --hash ] [ -o, --offset ] [ -l, --length ] file`

The same as **digest**, but operates on local files.

`lls [ -R ] [ -l ] [ -S ] [ -r ] [ -p ] [ -z|+z ] [ file ...]`

The same as **ls**, but operates on local files.

`llsroots`

The same as **lsroots**, but operates on local files (when the local end has been opened to a VShell server).

`lmkdir` *`directory`*

The same as **mkdir**, but operates on local files.

`localopen` [*`user@`*] *`hostname`* [*`#port`*] [`-l`] [`--user=` *`USERNAME`*]

The same as **lopen**.

`lopen` [*`user@`*] *`hostname`* [*`#port`*] [`-l`] [`--user=` *`USERNAME`*]

Tries to connect the local end to the host *`hostname`*. If this is successful, **lls** and friends will operate on the file system on that host.

Options:

`-l`

Connects the local end to the file system of the SFTP client host (which does not require a server). This is also the default state when no **lopen** commands have been given.

`--user`

Defines the user in the connection to be *`USERNAME`*.

`locsite` [`none` | *`name1=value1 name2=value2`* ...]

The same as **site**, but operates on local files and data sets.

`lpwd`

Prints the name of the current local working directory.

`lreadlink` *`path`*

The same as **readlink**, but operates on local files.

`lrename` *`oldfile newfile`*

The same as **rename**, but operates on local files.

`lrm` [`options` ...] *`file`* ...

The same as **rm**, but operates on local files.

`lrmdir` *`directory`*

The same as **rmdir**, but operates on local files.

`ls` [`-R`] [`-l`] [`-S`] [`-r`] [`-p`] [`-z|+z`] [*`file`* ...]

Lists the names of files on the remote server. For directories, contents are listed. If no arguments are given, the contents of the current working directory are listed.

Options:

`-R`

Directory trees are listed recursively. By default, subdirectories of the arguments are not visited.

`-l`

Permissions, owners, sizes and modification times are also shown (long format).

`-S`

Sorting is done based on file sizes (default: alphabetical sorting).

`-r`

The sort order is reversed.

`-p`

Only one page of listing is shown at a time.

`-z`

The client generates the long output.

`+z`

The long output supplied by the server is used, if available (alias for option `-l`).

`lsite` [`none` | *`name1=value1 name2=value2`* ...]

The same as **site**, but operates on local files and data sets.

`lsroots`

Dumps the virtual roots of the server. (This is a VShell extension. Without this you cannot know the file system structure of a VShell server.)

`lsymlink` *`targetpath linkpath`*

The same as **symlink**, but operates on local files.

`mget` [`options` ...] *`file`* ...

Synonymous to **get**, except for the SFTP compatibility mode option *ftp*. The SFTP compatibility mode options for command **mget**:

`tectia`

The **sftpg3** client transfers files recursively from the current directory and all its subdirectories.

`ftp, openssh`

Directories are not copied. Only regular files and symbolic links are copied.

`mkdir` *`directory`*

Tries to create the directory specified in *`directory`*.

`mput` [`options` ...] *`file`* ...

Transfers the specified files from the local end to the remote end. Options and semantics are the same as for **mget**, otherwise synonymous to **put**.

`open` [*`user@`*] *`hostname`* [*`#port`*] [`-l`] [`--user=`*`USERNAME`*]

Tries to connect the remote end to the host *`hostname`*.

Options:

`-l`

Connects the remote end to the file system of the SFTP client host (which does not require a server).

`--user`

Defines the user in the connection to be *`USERNAME`*.

`pause` [*`seconds`*]

Pauses batch file execution for *`seconds`* seconds, or if *`seconds`* is not given until **ENTER** is pressed.

`put` [`options` ...] *`file`* ...

Transfers the specified files from the local end to the remote end. Options and semantics are the same as for **get**.

`pwd`

Prints the name of the current remote working directory.

`quit`

Quits the application.

`readlink` *`path`*

Provided that *`path`* is a symbolic link, shows where the link is pointing to.

`rename` *`oldfile`* *`newfile`*

Tries to rename the *`oldfile`* to *`newfile`*. If *`newfile`* already exists, the files are left intact.

`rm` [`-I, --interactive`] [`-r, --recursive`] *`file`* ...

Tries to delete a file or directory specified in *`file`*.

Options:

`-I, --interactive`

Prompts whether to remove a file or directory (does not work with batch mode).

`-r, --recursive`

Directories are removed recursively.

`rmdir` *`directory`*

Tries to delete the directory specified in *`directory`*. This command removes the directory only if it is empty and has no subdirectories.

`set [ defaults | [ --commands=`*`name1,name2,...`*` exit-value=`*`VALUE`*` ] | `*`option1=value1 option2=value2 ...`*` ]`

Sets the default values for various parameters. The `set` command takes the following options:

`defaults`

Sets the parameters to be system defaults.

`checksum` [`=yes` | `no` | `md5` | `sha1` | `md5-force` | `sha1-force` | `checkpoint`]

Uses MD5 or SHA-1 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than *`buffer_size_bytes`* are not checked. Use `md5-force` or `sha1-force` with small files. The default is `md5` (in z/OS the default is `no`). Use checkpointing when transferring large files one by one.

`compatibility-mode` [`=tectia` | `ftp` | `openssh`]

Defines what mode of recursiveness is used in the file transfer:

`tectia`

The **sftpg3** client transfers files recursively from the current directory and all its subdirectories. This is the default mode.

`ftp`

A single file is transferred, and no subdirectories are copied.

`openssh`

Only regular files and symbolic links from the specified directory are copied, and no subdirectories are copied.

`compressions` [`=none` | `zlib`]

Defines whether compression is used in file transfer:

`none`

Compression is not used. This is the default.

`zlib`

Enables zlib compression in file transfer.

`exit-value=` *`VALUE`*

Defines the exit value of **sftpg3** in batch mode in case of an error. The value must be between 0 and 255. If `exit-value` is set to something else than 0 and the `--commands` parameter is not used, batch execution terminates when the first error occurs.

Example 1: If the rename command in this batch job fails, **sftpg3** will stop and return exit value "6":

```
open user@host
set exit-value=6
rename file file2
<next command in batch job>
quit
```

Example 2: If you want to ignore a possible failure of a specific command and return exit value "0" independent of the actual result of the operation, use `set exit-value=0` after the command. This example batch job ignores possible failure in renaming a file:

```
open user@host
rename file file2
set exit-value=0
<next command in batch job>
quit
```

`--commands=` *`name1,name2,...`* `exit-value=` *`VALUE`*

This option makes an **sftpg3** batch job abort when any of the specified commands fail. When a command that is *not* specified with this option fails, the batch job execution continues, and the exit value of the batch job is set to the one defined with `exit-value`. Note that if `exit-value=0`, the exit value of the failed command will be returned.

Example 3: When **sftpg3** is running in batch mode, it will abort execution if a **put**, **get**, or **ls** command fails. If any other command (with the exceptions mentioned below) fails, execution will continue until the end of the batch file. In both cases value "3" will be returned:

```
set --commands=put,get,ls exit-value=3
```

Example 4: When **sftpg3** is running in batch mode, it will abort execution when a **put** or **get** command fails. If any other command (with the exceptions mentioned below) fails, execution will continue. In any case the original exit value of the last failed command will be returned:

```
set --commands=put,get exit-value=0
```

Exceptions: When `exit-value` is set for specific commands with the `--commands` option, also the following situations will cause the batch job execution to abort:

- A **cd** command resulting in an error
- Any invalid command
- `Authentication failed` error
- `Unable to connect to server` error
- `Connection aborted` error

By default (`set defaults`), in case of errors, **sftpg3** does not stop but instead will continue executing and return the last error message.

Invalid commands added in `--commands` will be ignored.

`overwrite` [`=yes` | `no`]

Decides whether to overwrite existing destination file(s) (default: `yes`).

`progress-display` [`=bar` | `line` | `no`]

Chooses the mode of displaying the progress during a file transfer operation. The default is `bar`, which shows a progress bar. Option `line` shows the progress information according to the settings made with the `progress-line-format` option. Option `no` disables progress display.

`progress-line-format=` *`FORMAT_STRING`*

Chooses what information will be shown on the progress line. Use this option when `--progress-display=line`. See the definitions of contents options in command: `get --progress-line-format`.

`progress-line-interval=` *`seconds`*

Defines how often the progress information is updated in line mode. The interval is given in seconds, and the default is 60 seconds.

`summary-display` [`=no` | `yes` | `simple` | `bytes`]

Chooses the style of the file transfer summary data to be displayed after a file transfer operation. With the summary display, the progress bar data is also displayed by default. Do not use this option with `--statistics`.

See the options described for command: `get --summary-display`

`summary-format=` *`FORMAT_STRING`*

Chooses the format and the contents of the summary. You can use this option when `--summary-display=yes`. Do not use this option with `--statistics`.

See the definitions of contents options in command: `get --summary-format`

`streaming` [`=yes` | `no` | `force` | `ext`]

Uses streaming in file transfer if the server supports it. Files smaller than *`buffer_size_bytes`* are not transferred using streaming. Use `force` with small files. Default: `yes`

Use `ext` with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The `streaming=ext` option requires also the `checksum=no` option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

`setext` [*`extension`* ...]

Sets the file extensions that will be ASCII in the auto transfer mode. Normal zsh-fileglob regexps can be used in the file extensions.

`setperm` *`fileperm`* [:*`dirperm`*]

Sets the default file or directory permission bits for upload. (Prefix *`fileperm`* with `p` to preserve permissions of existing files or directories.)

`sget` [`options` ...] *`srcfile`* [*`dstfile`*]

Transfers a single specified file from the remote end to the local end under the filename defined with *`dstfile`*. Directories are not copied. No wildcards can be used. Options are the same as for **get**.

`site` [`none` | *`name1=value1 name2=value2`* ...]

Sets the file and data set parameters for the remote host. Parameters can be entered either one by one, or several parameters can be delimited by spaces or commas. Both long parameters and abbreviations can be used. When run without arguments, the **site** command outputs the list of entered parameters. Setting `none` resets all parameters.

The available parameters are:

- `AUTOMOUNT=YES|NO|IMMED`
- `[NO]AUTOMOUNT|[NO]AUTOM`
- `AUTORECALL=YES|NO`
- `[NO]AUTORECALL|[NO]AUTOR`
- `BLKSIZE|B|BLOCKSI=` *`size`*
- `BLOCKS|BL`
- `CONDDISP|CO=CATLG|UNCATLG|KEEP|DELETE`

- CYLINDERS|CY
- DATACLAS|DA= *class*
- DATASET_SEQUENCE_NUMBER|SEQNUM= *number*
- DEFER|DE=YES|NO
- [NO]DEFER|DE
- DIRECTORY_SIZE|M|DI|DIRSZ= *size*
- EXPIRY_DATE|EXPDT= *yyddd*|*yyyyddd*
- FILE_STATUS|STATUS=NEW|MOD|SHR|OLD
- FILETYPE|FILET=SEQ|JES
- FIXRECFM|FI= *length*
- JOB_ID|JESID= *ID*
- JOB_OWNER|JESO= *name*
- JOBNAME|JESJOB= *name*
- KEYLEN|KEYL= *length*
- KEYOFF|KEYO= *offset*
- LABEL_TYPE|LABEL=NL|SL|NSL|SUL|BLP|LTM|AL|AUL
- LIKE= *like*
- LRECL|R|LR= *length*
- MGMTCLAS|MG= *class*
- NORMDISP|NOR=CATLG|UNCATLG|KEEP|DELETE
- PRIMARY_SPACE|PRI= *space*
- PROFILE|P|PROF= *profile*
- RECFM|O|REC= *recfm*
- RECORD_TRUNCATE|U|TRUN=YES|NO
- [NO]TRUNCATE|[NO]TRU|[NO]TRUN
- RETENTION_PERIOD|RET= *days*
- SECONDARY_SPACE|SE|SEC= *space*
- SIZE|L= *size*

- SPACE_RELEASE|RLSE=YES|NO
- SPACE_UNIT|SU=BLKS|TRKS|CYLS|AVGRECLEN
- SPACE_UNIT_LENGTH|SUL= *length*
- STAGING|S|STAGE=YES|NO
- STORCLAS|ST= *class*
- SVC99_TEXT_UNITS|SVC99= *string*
- TRACKS|TR
- TRAILING_BLANKS|TRAIL=YES|NO
- [NO]TRAILINGBLANKS|[NO]TRAI|[NO]TRAIL
- TRANSFER_CODESET|C|CODESET= *codeset*
- TRANSFER_FILE_CODESET|D|FCODESET= *codeset*
- TRANSFER_FILE_LINE_DELIMITER|J|FLDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL
- TRANSFER_FORMAT|F|FORMAT=LINE|STREAM|RECORD
- TRANSFER_LINE_DELIMITER|I|LDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL
- TRANSFER_MODE|X|MODE=BIN|TEXT
- TRANSFER_TRANSLATE_DSN_TEMPLATES|A|XDSNT= *templates*
- TRANSFER_TRANSLATE_TABLE|E|XTBL= *table*
- TYPE|T=PS|PO|PDS|POE|PDSE|GDG|HFS|VSAM|ESDS|KSDS|RRN
- UNIT|UN= *unit*
- UNIT_COUNT|UC|UNC= *number*
- UNIT_PARALLEL|UNP=YES|NO
- VOLUME_COUNT|VC|VOLCNT= *number*
- VOLUMES|VO|VOL= *vol1+vol2+...*

`sput` [`options` ...] *`srcfile`* [*`dstfile`*]

Transfers a single specified file from the local end to the remote end under the filename defined with *`dstfile`*. Directories are not copied. No wildcards can be used. Options are the same as for **get**.

`sunique` [`on`] [`off`]

Stores files with unique names. If no option is specified, the command toggles the state of 'sunique'.

In case more than one of the transferred files have the same name, this feature adds a sequential number to the end of the repeated filename, for example: `file.name`, `file.name1`, and `file.name2`.

`symlink` *`targetpath linkpath`*

Creates symbolic link *`linkpath`*, which will point to *`targetpath`*.

`type` [`ascii` | `auto` | `binary` | `default`]

Sets file transfer type. If type is not specified, the current file transfer type is displayed.

`ascii`

Transfer file in ascii mode. See **ascii** for more information.

`auto`

Transfer file in auto mode. See **auto** for more information.

`binary`

Transfer file in binary mode. See **binary** for more information.

`default`

Transfer file in binary mode. This mode is identical to `binary` except that when the server is Tectia Server for IBM z/OS, no extra parameters are specified for the server. See **binary** for more information.

`verbose`

Enables verbose mode (identical to the **debug 2** command). You may later disable verbose mode by **debug disable**.

`help` [*`topic`*]

If *`topic`* is not given, lists the available topics. If *`topic`* is given, outputs available online help about the topic.

`helpall`

Outputs available online help about all topics.

Command Interpretation

sftpg3 understands both backslashes (\) and quotation marks (""") on the command line. A backslash can be used for ignoring the special meaning of any character in the command-line interpretation. It will be removed even if the character it precedes has no special meaning.

When specifying filenames that contain spaces, enclose them in quotation marks.

Note

Commands **get .** and **put .** will get or put every file in the current directory and possibly they overwrite files in your current directory.

sftpg3 supports wildcard characters (also known as glob patterns) given to commands **chmod**, **lchmod**, **ls**, **lls**, **rm**, **lrm**, **get**, and **put**.

Command-Line Editing (Unix)

On Unix, the following key sequences can be used for command-line editing:

Ctrl-Space

Set mark.

Ctrl-A

Go to the beginning of the line.

Ctrl-B

Move the cursor one character to the left.

Ctrl-D

Erase the character to the right of the cursor, or exit the program if the command line is empty.

Ctrl-E

Go to the end of the line.

Ctrl-F

Move the cursor one character to the right.

Ctrl-H

Backspace.

Ctrl-I

Tab.

Ctrl-J

Enter.

Ctrl-K

Delete the rest of the line.

Ctrl-L

Redraw the line.

Ctrl-M

Enter.

Ctrl-N

Move to the next line.

Ctrl-P

Move to the previous line.

Ctrl-T

Toggle two characters.

Ctrl-U

Delete the line.

Ctrl-W

Delete a region (the region's other end is marked with Ctrl-Space).

Ctrl-X

Begin an extended command.

Ctrl-Y

Yank deleted line.

Ctrl-_

Undo.

Ctrl-X Ctrl-L

Lower case region.

Ctrl-X Ctrl-U

Upper case region.

Ctrl-X Ctrl-X

Exchange cursor and mark.

Ctrl-X H

Mark the whole buffer.

Ctrl-X U

Undo.

Esc Ctrl-H

Backwards word delete.

Esc Delete

Backwards word delete.

Esc Space

Delete extra spaces (leaves only one space).

Esc <

Go to the beginning of the line.

Esc >

Go to the end of the line.

Esc @

Mark current word.

Esc A

Go back one sentence.

Esc B

Go back one word.

Esc C

Capitalize current word.

Esc D

Delete current word.

Esc E

Go forward one sentence.

Esc F

Go forward one word.

Esc K

Delete current sentence.

Esc L

Change current word to lower case.

Esc T

Transpose words.

Esc U

Change current word to upper case.

Delete

Backspace.

Filename Support

Different operating systems allow different character sets in filenames. On Unix, some of the special characters are allowed in filenames, but on Windows, the following characters are not allowed:

```
\/ : * ? " < > |
```

The **sftpg3** command-line tool (both as an interactive and in a batch file) follows the syntax and semantics of Unix shell command-line also on the Windows platform, except that the escape character is ~ (tilde).

When you transfer files that have special characters in the filename (for example `unixfilename*?".txt`) from a Unix server to Windows, you need to provide the files with new names that are acceptable on Windows.

The **sftpg3** command-line client includes two versions of the **get** command:

The **get** command can be used to transfer several files at the same time, but it is not possible to define target filenames. Note that if there are special characters in the filenames, you need to rename the files already on Unix so that the names are acceptable also on Windows.

The **sget** command is used to transfer one file at a time, and it allows you to define a new name for the destination file. Use it to make the name acceptable on Windows. The command sequence is as follows:

```
$ sftpg3
```

```
sftp> open user@server
```

```
sftp> sget "file*name.txt" windowsfilename.txt
```

Escaping special characters

In the **sftpg3** command, the following characters have a special meaning, and they need to be escaped in commands that take filenames as arguments:

* asterisk is a wildcard character for any number of any characters

? question mark is a wildcard for any single character

" quotation marks are placed around strings that are to be taken 'as is'

**** backslash is an escape character on Unix

~ tilde is an escape character on Windows

The escape character tells the **sftpg3** command to treat the next character "as is" and not to assume any special meaning for it. The escape character is selected according to the operating system of the local machine.

Note that the **** and **~** characters are special characters themselves, and if they are present in the filename, an escape character must be placed in front of them,too. Therefore, if you need to enter a filename containing **** in Unix or **~** in Windows to any of the **sftpg3** commands, add the relevant escape character to it:

**** on Unix

~~ on Windows

When a filename or part of a filename is placed within the quotation marks **"**, the **sftpg3** command interprets the quoted part 'as is', and none of the characters within the quote are interpreted as wildcards or as any other special characters.

However, on Unix a quotation mark **"** can also be part of a filename. If you need to enter the **"** character in a filename, you must add the escape character in front of it both on Unix and on Windows.

For example, to enter a file named `file-"name".txt` into a command on Windows, enter the following command:

```
sftp> sget "file-~"name~".txt" filename.txt
```

See the examples below to learn how the escape characters are used in the Tectia **sftpg3** commands, and how to enter filenames with special characters in different operating systems.

Examples of filenames in the **sftpg3** commands:

The following filenames are valid in Unix, but they need escape characters in the commands:

```
file|name.txt
file-"name".txt
file?name.txt
file*name.txt
file\name.txt
file - name.txt
file~name.txt
```

When using the **sftpg3** command-line tool on Unix, enter the above mentioned filenames in the following formats:

```
file\|name.txt     or  "file|name.txt"
file-\"name\".txt  or  "file-\"name\".txt"
file\?name.txt     or  "file?name.txt"
```

```
file\*name.txt     or  "file*name.txt"
file\\name.txt     or  "file\\name.txt"
file\ -\ name.txt  or  "file - name.txt"
file~name.txt      or  "file~name.txt"
```

Example commands on Unix:

```
sftp> get "file*name.txt"
```

```
sftp> sget "file*name.txt" newfilename.txt
```

When using the **sftpg3** command on Windows, enter the above mentioned Unix filenames in the following formats:

```
file~|name.txt     or  "file|name.txt"
file-~"name~".txt  or  "file-~"name~".txt"
file~?name.txt     or  "file?name.txt"
file~*name.txt     or  "file*name.txt"
file~\name.txt     or  "file\name.txt"
file~ -~ name.txt  or  "file - name.txt"
file~~name.txt     or  "file~~name.txt"
```

Example command sequence on Windows:

```
> sftpg3 open user@server
```

```
sftp> get "file name.txt"
```

```
sftp> sget "file*name.txt" filename.txt
```

Environment Variables

sftpg3 uses the following environment variables:

SSH_SFTP_BATCH_FILE=*startup_batch_file*

Defines the path to the **sftpg3** startup batch file. The file is run and the **sftpg3** commands defined in the file are executed each time **sftpg3** is started.

If this variable is not defined, **sftpg3** looks for a startup batch file named `ssh_sftp_batch_file` in the user-specific directory `$HOME/.ssh2/` on Unix or `%APPDATA%\SSH\` on Windows.

Note that if this variable is defined but the file is missing or cannot be accessed, **sftpg3** fails to start.

SSH_SFTP_CHECKSUM_MODE `=yes|no|md5|sha1|sha256|sha512|md5-force|sha1-force|sha256-force|sha512-force|checkpoint`

Defines the setting for comparing checksums. For more information on the available values, see **checksum** .

SSH_SFTP_SHOW_BYTE_COUNT `=yes|no`

If this variable is set to `yes`, the number of transferred bytes is shown after successful file transfer. Also the names of source and destination files are shown. The default is `no`.

SSH_SFTP_STATISTICS `=yes|no|simple`

If this variable is set to `yes` (default), normal progress bar is shown while transferring the file. If it is set to `no`, progress bar is not shown. If it is set to `simple`, file transfer statistics are shown after the file has been transferred.

UTF8_MODE `=0|1`

If this variable is set to `0`, the default charset mode UTF-8 is disabled. To force UTF-8 charset mode for file names, set the variable to `1`.

Exit Values

sftpg3 returns the following values based on the result of the operation:

```
0     Operation was successful.
1     Internal error.
2     Connection aborted by the user.
3     Destination is not a directory, but a directory was specified by the user.
4     Connecting to the host failed.
5     Connection lost.
6     File does not exist.
7     No permission to access file.
8     Undetermined error from sshfilexfer.
11    Some non-fatal errors occured during a directory operation.
101   Wrong command-line arguments specified by the user.
```

In batch mode, **sftpg3** returns the value 0 only if no errors occurred during the execution. A failure to change the current working directory, a failure to establish a connection, or a connection loss during batch operation cause **sftpg3** to abort. Other errors are reported to stderr and the last error value is returned as the exit value of the **sftpg3** process.

Examples

Open a **sftpg3** session with the remote end connected to the server defined in the connection profile *`profile1`* in the `ssh-broker-config.xml` file (the local end is initially connected to the file system of the SFTP client host):

```
$ sftpg3 profile1
```

Run **sftpg3** in batch mode:

```
$ sftpg3 -B batch.txt
```

Example contents of the batch file *`batch.txt`* are shown below. Non-interactive authentication methods are used and the server host keys have been stored beforehand:

```
lopen user@unixserver.example.com
open user@winserver.example.com
binary
lcd backup
```

```
cd c:/temp
get --force-lower-case Testfile-X.bin
lchmod 700 testfile-x.bin
digest -H sha256 Testfile-X.bin
ldigest -H sha256 testfile-x.bin
quit
```

The example batch file opens the local end of the connection to a Unix server and the remote end to a Windows server, and sets the transfer mode to binary. It changes to local directory *backup* and remote directory *C:\Temp*, and copies a file from the remote directory to the local directory. The filename is changed to lower-case characters (*testfile-x.bin*). After transfer, the file permissions are changed to allow the user full rights and others no rights, and additional SHA-2 hash are calculated of the source and destination files for further verification.

ssh-translation-table

ssh-translation-table — Secure Shell Translation Table

Synopsis

`ssh-translation-table` [`options` ...]
[`filename`]

Description

ssh-translation-table (**ssh-translation-table.exe** on Windows) is a utility program that generates translation tables for coded character set (CCS) conversions. **ssh-translation-table** stores the translation table in *`filename`*. If *`filename`* is not given, **ssh-translation-table** writes the translation table to standard output.

Options

The following options are available:

`-b, --binary`

Use the z/OS-specific binary file format.

`-f, --from=` *`CODESET`*

Specify the source code set of the inbound conversion, which is also the target code set of the outbound conversion. The default value is *`ISO8859-1`*. For example:

```
--from ISO8859-15
```

`-t, --to=` *`CODESET`*

Specify the target code set of the inbound conversion, which is also the source code set of the outbound conversion. The default value is *`IBM-1047,swaplfnl`* if the underlying implementation is ICU, otherwise *`IBM-1047`*. For example:

```
--to IBM-037
```

`-l, --list-charsets`

List available character sets. Note that all character sets are not single byte character sets. Only single byte character sets can be used.

`-D, --debug=` *`LEVEL`*

Sets the debug level. *`LEVEL`* is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

`-h, --help`

Displays a short summary of command-line options and exits.

Translation Table

A translation table is a file containing two tables describing the character conversion, the inbound table and the outbound table. Each table consists of 256 target values.

In Tectia File Transfer, the inbound table is used when converting data from the line to the data set. The outbound table is used when converting data from a file and sending the data out on the line.

The binary format, which is z/OS specific, consists of three 256 byte fields. The first is a comment in EBCDIC, which is ignored in the conversion software, the second is the inbound table and the third is the outbound table.

The text format can have interspersed comments. The target values are in hexadecimal.

A table is a list of 256 values represented as two hexadecimal characters (from 00 to FF). The position of the value is the index for conversion. The first position, i.e. position 00, represents the converted value for byte value of 0.

The hexadecimal values in the tables are case-insensitive. So values 0a and 0A are the same. Also, it is possible to add comments into the file. The comment starts with character '#'. Everything after that until end of line is treated as comment and ignored. Also all white spaces are ignored.

Note

Only single byte translations are supported with translation tables.

Here is an example translation table generated with command **ssh-translation-table**:

```
## SSH TRANSLATION TABLE FILE FORMAT VERSION 1.0
#####################################################################
#
# This file is an example translation table that can be used to
# translate data from 'ISO8859-1' to 'IBM-1047,swaplfnl' while reading
# from a file or from 'IBM-1047,swaplfnl' to 'ISO8859-1' while writing
# to a file.
#
# The format of translation table file is following:
#
# - White spaces are ignored.
# - Everything after '#' character until end of line is a comment
#   that is ignored.
# - The first table is used when writing data to a file.
# - The second table is used when reading data from a file.
# - Both tables must exist.
# - Table is a simple hexadecimal representation of the
#   translation. Each value is represented as two hexadecimal
#   characters. The first line gives the values in table
#   positions 0-15 (00-0F), the second line 16-31 (10-1F)
#   and so on.
#
# Note: Only single byte translations are supported.
#
#####################################################################
```

```
# Inbound (network to file) translation table:
# IBM-1047,swaplfnl -> ISO8859-1

#0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F
000102039C09867F978D8E0B0C0D0E0F #0
101112139D0A08871819928F1C1D1E1F #1
808182838485171B88898A8B8C050607 #2
909116939495960498999A9B14159E1A #3
20A0E2E4E0E1E3E5E7F1A22E3C282B7C #4
26E9EAEBE8EDEEEFECDF21242A293B5E #5
2D2FC2C4C0C1C3C5C7D1A62C255F3E3F #6
F8C9CACBC8CDCECFCC603A2340273D22 #7
D861626364656667686 9ABBBF0FDFEB1 #8
B06A6B6C6D6E6F707172AABAE6B8C6A4 #9
B57E737475767778797AA1BFD05BDEAE #A
ACA3A5B7A9A7B6BCBDBEDDA8AF5DB4D7 #B
7B414243444546474849ADF4F6F2F3F5 #C
7D4A4B4C4D4E4F505152B9FBFCF9FAFF #D
5CF7535455565758595AB2D4D6D2D3D5 #E
30313233343536373839B3DBDCD9DA9F #F

# Outbound (file to network) translation table:
# ISO8859-1 -> IBM-1047,swaplfnl
#
#0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F
00010203372D2E2F1605150B0C0D0E0F #0
101112133C3D322618193F271C1D1E1F #1
405A7F7B5B6C507D4D5D5C4E6B604B61 #2
F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F #3
7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6 #4
D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D #5
79818283848586878889919293949596 #6
979899A2A3A4A5A6A7A8A9C04FD0A107 #7
202122232425061728292A2B2C090A1B #8
30311A333435360838393A3B04143EFF #9
41AA4AB19FB26AB5BBB49A8AB0CAAFBC #A
908FEAFABEA0B6B39DDA9B8BB7B8B9AB #B
6465626663679E687471727378757677 #C
AC69EDEEEBEFECBF80FDFEFBFCBAAE59 #D
4445424643479C485451525358555657 #E
8C49CDCECBCFCCE170DDDEDBDC8D8EDF #F

# EOF
```

Note

When ICU libraries are used for generating ASCII to EBCDICtranslation tables, `,swaplfnl` must be added to the EBCDIC codepage name so that ASCII line feed characters (0A) are correctly translated to EBCDIC newline characters (15).

In order to create a custom translation table, first create a translation table with **ssh-translation-table** and then edit it with any text editor.

Environment Variables

`SSH_CHARSET_CONV`

The full pathname of the Tectia conversion DLL. Only required if **ssh-translation-table** or the conversion DLL are not in the installation directories. Here is an example of the pathname:

```
SSH_CHARSET_CONV=/opt/tectia/lib/shlib/i18n_iconv.so
```

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 Client and store the key to global configuration `hostkeys` directory:

```
$ ssh-keyfetch -a --output-directory /etc/ssh2/hostkeys
Accepted newhost 22 testuser /etc/ssh2/hostkeys/key_22_anotherhost.pub
bydop-mulym-zegar-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
      407596441316907505326652662186964660603777993580367354759022
      576086098562919363963470926690162744258451983124575595926849
      551903
    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 C Egrep Syntax

The Tectia tunneling filter rules can be matched to hostname or IP address patterns specified using the **egrep** syntax. In addition, regular expressions can be used in selectors when specifying ranges of values. The egrep syntax is explained in this section.

C.1 Egrep Patterns

The escape character is a backslash (\). You can use it to escape meta characters to use them in their plain character form.

In the following examples literal 'E' and 'F' denote any expression, whether a pattern or a character.

(

Start a capturing subexpression.

)

End a capturing subexpression.

E|F

Disjunction, match either E or F (inclusive). E is preferred if both match.

E*

Act as Kleene star, match E zero or more times.

E+

Closure, match E one or more times.

E?

Option, match E optionally once.

.

Match any character except for newline characters (\n, \f, \r) and the `NULL` byte.

E{n}

Match E exactly n times.

E{n,} or E{n,0}

Match E n or more times.

E{,n} or E{0,n}

Match E at most n times.

E{n,m}

Match E no less than n times and no more than m times.

[

Start a character set, see Section C.3.

$

Match the empty string at the end of the input or at the end of a line.

^

Match the empty string at the start of the input or at the beginning of a line.

C.2 Escaped Tokens for Regex Syntax Egrep

\0n..n

The literal byte with octal value n..n.

\0

The `NULL` byte.

\[1-9]..x

The literal byte with decimal value `[1-9]..x`.

\xn..n or \0xn..n

The literal byte with hexadecimal value n..n.

\<

Match the empty string at the beginning of a word.

\>

Match the empty string at the end of a word.

\b

Match the empty string at a word boundary.

\B

Match the empty string provided it is not at a word boundary.

\w

Match a word-constituent character, equivalent to [a:zA:Z0:9-].

\W

Match a non-word-constituent character.

\a

Literal alarm character.

\e

Literal escape character.

\f

Literal line feed.

\n

Literal new line, equivalent to C's \n so it can be more than one character long.

\r

Literal carriage return.

\t

Literal tab.

All other escaped characters denote the literal character itself.

C.3 Character Sets For Egrep

A character set starts with '[' and ends at non-escaped ']' that is not part of a POSIX character set specifier and that does not follow immediately after '['.

The following characters have a special meaning and need to be escaped if meant literally:

- (minus sign)

    A range operator, except immediately after '[', where it loses its special meaning.

^

    If immediately after the starting '[', denotes a complement: the whole character set will be complemented. Otherwise literal '^'.

[:alnum:]

    Characters for which 'isalnum' returns true .

[:alpha:]

    Characters for which 'isalpha' returns true .

[:cntrl:]

    Characters for which 'iscntrl' returns true .

[:digit:]

    Characters for which 'isdigit' returns true .

[:graph:]

    Characters for which 'isgraph' returns true .

[:lower:]

    Characters for which 'islower' returns true .

[:print:]

    Characters for which 'isprint' returns true .

[:punct:]

    Characters for which 'ispunct' returns true .

[:space:]

    Characters for which 'isspace' returns true .

[:upper:]

    Characters for which 'isupper' returns true .

[:xdigit:]

Characters for which 'isxdigit' returns true .

Example: `[[:xdigit:]XY]` is typically equivalent to `[0123456789ABCDEFabcdefXY]` .

It is also possible to include the predefined escaped character sets into a newly defined one, so `[\d\s]` matches digits and whitespace characters.

Also, escape sequences resulting in literals work inside character sets.

Appendix D Audit Messages

This appendix lists the audit messages generated by the Connection Broker.

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.

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)

Argument	Description
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: Connection Broker

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)
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

1122 Certificate_end_point_id_check_success
Level: informational
Origin: Connection Broker

End point identity check succeeded.

Default log facility: normal

Argument	Description
Server	Host name
Text	Explanatory message

1123 Certificate_end_point_id_check_warning
Level: informational
Origin: Connection Broker

Certificate end point identity check warning.

Default log facility: normal

Argument	Description
Server	Host name
Text	Warning message

1124 Certificate_end_point_id_check_failure
Level: informational
Origin: Connection Broker

Certificate end point identity check failure.

Default log facility: normal

Argument	Description
Server	Host name

Argument	Description
Text	Error message

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
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

Argument	Description
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

3000 Sft_client_start
Level: debug
Origin: Tectia Secure File Transfer clients

File transfer client program was started.

Default log facility: user

3001 Sftc_create_file
Level: debug
Origin: Tectia Secure File Transfer clients

A new file was created.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
File	Path to target file
Text	(Optional) error message and/or additional information

3002 Sftc_truncate_file
Level: debug
Origin: Tectia Secure File Transfer clients

A file was truncated.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program

Argument	Description
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
File	Path to file
Text	(Optional) error message and/or additional information

3003 Sftc_modify_file_attrs
Level: informational
Origin: Tectia Secure File Transfer clients

A file attribute was modified.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
File	Path to file
Text	(Optional) error message and/or additional information

3004 Sftc_delete_file
Level: notice
Origin: Tectia Secure File Transfer clients

A file was deleted.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
File	Path to file
Text	(Optional) error message and/or additional information

3005 Sftc_create_dir

Level: debug
Origin: Tectia Secure File Transfer clients

A directory was created.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
Dir	Path to directory
Text	(Optional) error message and/or additional information

3006 Sftc_remove_dir
Level: notice
Origin: Tectia Secure File Transfer clients

A directory was removed.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
Dir	Path to directory
Text	(Optional) error message and/or additional information

3007 Sftc_copy_dir_start
Level: notice
Origin: Tectia Secure File Transfer clients

Copying a directory initiated.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID

Argument	Description
OpID	Operation ID
Source Connection	Source connection
From	Path to source directory
Target Connection	Target connection
To	Path to target directory
Text	(Optional) error message and/or additional information

3008 Sftc_copy_dir_finished
Level: notice
Origin: Tectia Secure File Transfer clients

Copying a directory completed.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Source Connection	Source connection
From	Path to source directory
Target Connection	Target connection
To	Path to target directory
Duration	Duration of copying the directory
Files	Number of files that were copied
Data	Amount of data copied in bytes
Speed	Copy speed in KiB/s
Text	(Optional) error message and/or additional information

3009 Sftc_move_dir_start
Level: notice
Origin: Tectia Secure File Transfer clients

Moving a directory initiated.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Source Connection	Source connection

Argument	Description
From	Path to source directory
Target Connection	Target connection
To	Path to target directory
Text	(Optional) error message and/or additional information

3010 Sftc_move_dir_finished
Level: notice
Origin: Tectia Secure File Transfer clients

Moving a directory completed.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Source Connection	Source connection
From	Path to source directory
Target Connection	Target connection
To	Path to target directory
Duration	Duration of moving the directory
Files	Number of files that were moved
Data	Amount of data transferred in bytes
Speed	Transfer speed in KiB/s
Text	(Optional) error message and/or additional information

3011 Sftc_copy_file_start
Level: informational
Origin: Tectia Secure File Transfer clients

Copying a file initiated.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Source Connection	Source connection
From	Path to source file
Target Connection	Target connection

Argument	Description
To	Path to target file
Text	(Optional) error message and/or additional information

3012 Sftc_copy_file_finished
Level: notice
Origin: Tectia Secure File Transfer clients

Copying a file completed.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Source Connection	Source connection
From	Path to source file
Target Connection	Target connection
To	Path to target file
Duration	Duration
Data	Amount of data copied in bytes
Speed	Copy speed in KiB/s
Text	(Optional) error message and/or additional information

3013 Sftc_move_file_start
Level: informational
Origin: Tectia Secure File Transfer clients

Moving a file initiated.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Source Connection	Source connection
From	Path to source file
Target Connection	Target connection
To	Path to target file
Text	(Optional) error message and/or additional information

3014 Sftc_move_file_finished
Level: notice
Origin: Tectia Secure File Transfer clients

Moving a file completed.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Source Connection	Source connection
From	Path to source file
Target Connection	Target connection
To	Path to target file
Duration	Duration of moving the file
Data	Amount of data transferred in bytes
Speed	Transfer speed in KiB/s
Text	(Optional) error message and/or additional information

3015 Sftc_rename_file
Level: informational
Origin: Tectia Secure File Transfer clients

A file was renamed.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
File	Path to file
Text	(Optional) error message and/or additional information

3017 Sft_client_command
Level: debug
Origin: Tectia Secure File Transfer clients

File transfer client command.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Text	Command given on the command line or defined in a batch file

3018 Sftc_open_dir
Level: debug
Origin: Tectia Secure File Transfer clients

A directory was opened.

Default log facility: user

Argument	Description
Local username	Local user name
Program	Secure file transfer client program
Pid	Client process ID
OpID	Operation ID
Status	Result (SUCCESS or FAILED)
Connection	Target connection
Dir	Path to directory
Text	(Optional) error message and/or additional information

6000 Broker_client_connect
Level: informational
Origin: Connection Broker

A client connected to the Broker.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Process id
Local username	Local user name

6001 Broker_client_connect_failed
Level: warning
Origin: Connection Broker

A client attempted to connect unsuccessfully to the Broker.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Process id
Local username	Local user name
Text	Reason

6002 Broker_client_disconnect
Level: informational
Origin: Connection Broker

A client disconnected from the Broker.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Process id
Local username	Local user name

6004 Broker_exec_channel_open
Level: informational
Origin: Connection Broker

The Broker opened an exec channel.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process ID
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Command	Command
Text	Exec parameters
Channel Id	Channel ID
Session-Id	Session ID

6005 Broker_exec_channel_open_failed
Level: warning
Origin: Connection Broker

The Broker failed to open an exec channel for a client.

Default log facility: normal

Argument	Description
Client	Client name

Argument	Description
Pid	Client process ID
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Command	Command
Text	Exec parameters
Channel Id	Channel ID
Text	Reason
Session-Id	Session ID

6006 Broker_tunnel_open
Level: informational
Origin: Connection Broker

The Broker opened a tunnel for a client.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process ID
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Dst	Destination host
Dst Port	Destination port
Tunnel type	Tunnel type
Session-Id	Session ID

6007 Broker_tunnel_open_failed
Level: warning
Origin: Connection Broker

The Broker failed to open a tunnel for a client.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Dst	Destination host

Argument	Description
Dst Port	Destination port
Tunnel type	Tunnel type
Text	Reason
Session-Id	Session ID

6008 Broker_tunnel_listener_open
Level: informational
Origin: Connection Broker

The Broker opened a tunnel listener for a client.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process ID
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Listener	Listener host
Listener Port	Listener port
Dst	Destination host
Dst Port	Destination port
Tunnel type	Tunnel type
Text	Tunnel listener parameters
Session-Id	Session ID

6009 Broker_tunnel_listener_open_failed
Level: warning
Origin: Connection Broker

The Broker failed to open a tunnel listener for a client.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Server	Server host
Server Port	Server port
Remote username	Remote user name
Local username	Local user name
Listener	Listener host
Listener Port	Listener port
Dst	Destination host
Dst Port	Destination port

Argument	Description
Tunnel type	Tunnel type
Text	Tunnel listener parameters
Text	Reason
Session-Id	Session ID

6010 Broker_channel_fd_strip
Level: informational
Origin: Connection Broker

The Broker destroyed a channel object (and returned the underlying fd to the client).

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Text	Channel permanent?
Local username	Local user name
Session-Id	Session ID

6011 Broker_channel_fd_strip_failed
Level: warning
Origin: Connection Broker

The Broker failed to destroy a channel object (and return the underlying fd to the client).

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Text	Channel permanent?
Local username	Local user name
Text	Reason
Session-Id	Session ID

6012 Broker_channel_control
Level: informational
Origin: Connection Broker

The Broker sent a channel control message.

Default log facility: discard

Argument	Description
Client	Client name

Argument	Description
Pid	Client process ID
Channel Id	Channel ID
Command	Command
Args	Arguments
Local username	Local user name
Session-Id	Session ID

6013 Broker_channel_control_failed
Level: warning
Origin: Connection Broker

The Broker failed to send a channel control message.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Command	Command
Args	Arguments
Local username	Local user name
Text	Reason
Session-Id	Session ID

6014 Broker_channel_close
Level: informational
Origin: Connection Broker

The Broker closed a channel.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Exit Value	Exit value
Local username	Local user name
Session-Id	Session ID

6015 Broker_channel_close_failed
Level: warning
Origin: Connection Broker

The Broker failed to close a channel.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Local username	Local user name
Text	Reason

6018 Broker_server_version_request
Level: informational
Origin: Connection Broker

The Broker requested (and got) the server version.

Default log facility: discard

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Ver	Version
Local username	Local user name
Session-Id	Session ID

6019 Broker_server_version_request_failed
Level: warning
Origin: Connection Broker

The Broker failed to get the server version.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Local username	Local user name
Text	Reason
Session-Id	Session ID

6020 Broker_channel_process_exit
Level: informational
Origin: Connection Broker

Channel process exit request was successful.

Default log facility: discard

Argument	Description
Client	Client name

Argument	Description
Pid	Client process ID
Local username	Local user name
Session-Id	Session ID

6021 Broker_channel_process_exit_failed
Level: warning
Origin: Connection Broker

Channel process exit request failed.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Text	Reason
Local username	Local user name
Session-Id	Session ID

6025 Broker_connector_license_check_failed
Level: warning
Origin: Connection Broker

Connector license check failed.

Default log facility: normal

Argument	Description
Text	Error message
Session-Id	Session id

6026 Broker_server_rekey
Level: notice
Origin: Connection Broker

The Broker requested rekeying and it was successful.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Local username	Local user name
Session-Id	Session ID

6027 Broker_server_rekey_failed
Level: warning

Origin: Connection Broker

The Broker requested rekeying but it failed.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process ID
Channel Id	Channel ID
Local username	Local user name
Text	Reason
Session-Id	Session ID

6035 Broker_publickey_upload
Level: informational
Origin: Connection Broker

Public key is uploaded.

Default log facility: normal

Argument	Description
Client	Client name
Pid	Client process id
Local username	Local user name
Public key hash	Public key hash
Public key hash (SHA-256)	Public key hash (SHA-256)
Server	Server name
Server Port	Server port
Remote username	Remote user name
File name	Public key file name

6100 Broker_starting
Level: notice
Origin: Connection Broker

The Broker is starting.

Default log facility: normal

Argument	Description
Local username	Local user name

6101 Broker_start_failed
Level: warning
Origin: Connection Broker

Starting the Broker failed.

Default log facility: normal

Argument	Description
Local username	Local user name
Success | Error	Error code
Text	Error message

6102 Broker_running
Level: notice
Origin: Connection Broker

The Broker is running.

Default log facility: normal

Argument	Description
Local username	Local user name
Text	Message text

6104 Broker_stopping
Level: notice
Origin: Connection Broker

The Broker is stopping.

Default log facility: normal

Argument	Description
Local username	Local user name

6106 Broker_reconfig_started
Level: notice
Origin: Connection Broker

Reconfiguration started.

Default log facility: normal

Argument	Description
Local username	Local user name

6108 Broker_reconfig_finished
Level: notice
Origin: Connection Broker

Reconfiguration finished.

Default log facility: normal

Argument	Description
Local username	Local user name

Argument	Description
Success | Error	Error code

6114 Broker_config_deprecated_element
Level: warning
Origin: Connection Broker

The Broker config contains a deprecated element.

Default log facility: normal

Argument	Description
Text	Event description.

6200 Broker_tcp_connect
Level: informational
Origin: Connection Broker

Broker TCP connection attempt was successful.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Src Port	Source port
Local username	Local username

6201 Broker_tcp_connect_failed
Level: warning
Origin: Connection Broker

Broker TCP connection attempt failed.

Default log facility: normal

Argument	Description
Dst	Destination host
Dst Port	Destination port
Local username	Local username
NIO error	NIO error

6204 Broker_transport_connect
Level: informational
Origin: Connection Broker

A transport was connected through TCP.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Remote username	Remote username
Src Port	Source port
Local username	Local username
Session-Id	Session ID

6206 Broker_transport_gateway_connect
Level: informational
Origin: Connection Broker

A transport was connected through a gateway handle.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Remote username	Remote username
Local username	Local username
Session-Id	Session ID

6208 Broker_connection_connect
Level: informational
Origin: Connection Broker

The Broker got successfully a Secure Shell connection up.

Default log facility: discard

Argument	Description
Dst	Destination host
Dst Port	Destination port
Local username	Local user name
Remote username	Remote user name
Uses gateway?	Is this going through a gateway handle
Session-Id	Session ID

6209 Broker_connection_connect_failed
Level: warning
Origin: Connection Broker

The Broker failed to get a Secure Shell connection up.

Default log facility: normal

Argument	Description
Dst	Destination host

Argument	Description
Dst Port	Destination port
Local username	Local user name
Remote username	Remote user name
Uses gateway?	Is this going through a gateway handle
Session-Id	Session ID
Text	Error code

6210 Broker_connection_disconnect
Level: informational
Origin: Connection Broker

A Secure Shell connection initiated by the Broker was disconnected.

Default log facility: discard

Argument	Description
Local username	Local user
Session-Id	Session identifier
Dst	Destination host
Dst Port	Destination port
Remote username	Remote username

6211 Broker_unknown_hostkey_accepted
Level: warning
Origin: Connection Broker

* The Broker accepted an unknown hostkey without user interaction * because of configuration.

Default log facility: normal

Argument	Description
Text	Key digest
Dst	Destination host
Dst Port	Destination port
Local username	Local user name
Remote username	Remote user name
Text	SHA-256 key digest

6212 Broker_new_hostkey
Level: warning
Origin: Connection Broker

* First connection to a server or this server hostkey was never * saved before.

Default log facility: normal

Argument	Description
Text	Key digest

Argument	Description
Dst	Destination host
Dst Port	Destination port
Local username	Local user name
Remote username	Remote user name
Text	SHA-256 key digest

6213 Broker_hostkey_changed
Level: warning
Origin: Connection Broker

* Server hostkey is different than the saved hostkey.

Default log facility: normal

Argument	Description
Text	Key digest
Dst	Destination host
Dst Port	Destination port
Local username	Local user name
Remote username	Remote user name
Text	SHA-256 key digest

6214 Broker_hostkey_rotation
Level: informational
Origin: Connection Broker

Client has added or removed host key from known hosts due to hostkey advertisement from the server.

Default log facility: normal

Argument	Description
Text	Event description.
Dst	Destination host
Dst Port	Destination port
Local username	Local user name
Text	SHA-1 fingerprint.
Text	SHA-256 key digest

6301 Broker_userauth_failure
Level: warning
Origin: Connection Broker

User authentication failed.

Default log facility: normal

Argument	Description
Text	Reason

Argument	Description
Session-Id	Session identifier

6302 Broker_userauth_method_success
Level: informational
Origin: Connection Broker

A user authentication method succeeded.

Default log facility: discard

Argument	Description
Text	Authentication method
Session-Id	Session identifier

6303 Broker_userauth_method_failure
Level: warning
Origin: Connection Broker

A user authentication method failed.

Default log facility: discard

Argument	Description
Text	Authentication method
Text	Reason
Session-Id	Session identifier

6401 Connector_filter_rule
Level: informational
Origin: Connection Broker

FTP_CAPTURE not tunneling

Default log facility: discard

Argument	Description
Connector	Connector action
Dst	Address
Dst Port	Port

6402 Capture_init_info
Level: informational
Origin: Connection Broker

Capture failed to start

Default log facility: discard

Argument	Description
Text	Error message

Appendix E Default and Supported SSH Algorithms

This section describes the SSH algorithms supported by Tectia products.

E.1 Ciphers

Table E.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 E.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	

E.2 Key-Exchange Algorithms

Table E.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 E.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)	•

E.3 Message-Authentication Codes

Table E.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 E.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)	

E.4 Host-Key and Public Key Signature Algorithms

Table E.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 E.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 F Removing OpenSSL from Tectia Client

F.1 Background Information

F.1.1 Should I Remove the OpenSSL Library?

When Tectia Client is not used in the FIPS compliant mode (for more information, see Section 3.6), 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 Client 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 Client 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 Client 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 Client 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 Section 3.6).

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 Client** and click **Change**.
3. In the installer click **Modify**.
4. Select **Tectia Client** > **FIPS** optional module and change it 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 Client 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 Client version and build numbers, for example `7.0.0.123`.)
- `<INSTALLDIR>\SSH Tectia AUX\fips\fips.dll`
- `<INSTALLDIR>\SSH Tectia AUX\fips\openssl.cnf`
- `<INSTALLDIR>\SSH Tectia AUX\libcrypto-3.dll`
- `<INSTALLDIR>\SSH Tectia Broker\libcrypto-3.dll`
- `<INSTALLDIR>\SSH Tectia Client\libcrypto-3.dll`

Appendix G 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

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X