Tectia Server 6.6 for IBM z/OS

Administrator Manual

06 July 2022

Tectia Server 6.6 for IBM z/OS : Administrator Manual

06 July 2022

Copyright © 2007–2022 SSH Communications Security Corporation

Table of Contents

Chapter 1 About This Document

This document contains instructions on installing and using Tectia Server for IBM z/OS on IBM mainframes.

To fully utilize the information presented in this document, you should be familiar with Unix System Services (USS) of z/OS and Unix concepts in general.

This document contains the following information:

- Installing Tectia Server for IBM z/OS

- Getting started with Tectia Server for IBM z/OS

- Server configuration settings

- Server and user authentication

- System administration

- Secure file transfer using SFTP

- Setting up transparent FTP security

- Secure Shell tunneling

- Troubleshooting

- Appendices

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

Tectia Server for IBM z/OS User Manual gives instructions on using the Tectia client tools on z/OS for secure system administration and secure file transfer.

Tectia Server for IBM z/OS Quick Start Guide gives instructions for getting started with Tectia Server for IBM z/OS on IBM mainframes.

The *Tectia man pages* describe the commands and configuration file options in detail. The man pages are located in the man subdirectory of the Tectia installation directory. The server man pages are also contained in Appendix A of this manual.

In addition, the following documents will aid in understanding the Secure Shell protocol and using the product:

• general literature about Secure Shell, e.g. *SSH, The Secure Shell: The Definitive Guide* by O'Reilly

• Secure Shell RFCs 4250-4256 at http://www.ietf.org/rfc/

1.1 Introduction to Tectia Server for IBM z/OS

Tectia Server for IBM z/OS is a client/server security solution based on the Secure Shell protocol. It provides secure file transfer, secure shell access, remote shell command execution, and TCP and FTP tunneling. Together with Tectia Client and ConnectSecure on Windows, it provides transparent secure TN3270 connections.

The server component of Tectia Server 6.6 for IBM z/OS is based on Tectia Server 4.3 but it has the same advanced file transfer features as Tectia Server 6.6 on Unix platforms.

The client components of Tectia Server 6.6 for IBM z/OS are based on Tectia Client 6.6 and they support the same advanced file transfer features as Tectia ConnectSecure 6.6 on Unix platforms.

Tectia Server 6.6 for IBM z/OS runs on z/OS (the supported z/OS versions are listed in Section 2.1.1). It is installed to z/OS Unix System Services (USS), but it can be run in the native environment (hereafter MVS) and process files of the native file system.

1.2 System Authorization Facility

Tectia Server for IBM z/OS uses the System Authorization Facility (SAF) interface to the system security product and will thus work together with RACF, ACF2, and Top Secret. ACF2 and Top Secret are products from Computer Associates. This document and sample files show RACF commands.

1.3 Sample Files

After installation, Tectia Server for IBM z/OS includes the following samples that complement this document:

• `<HLQ>.V6611.SAMPLIB`: a PDS containing JCL jobs for a variety of SSH tasks

• `/opt/tectia/doc/zOS/samples`: a directory containing USS environment settings

Before running the samples, you should adapt them to your environment. For example, change the user, file, and machine names to those of your system.

The tasks described by the samples can of course be accomplished by other means, for example by using the shell.

1.4 Documentation Conventions

The following typographical conventions are used in Tectia documentation:

Table 1.1. Documentation conventions

Convention	Usage	Example
Bold	Tools, menus, GUI elements and commands, command-line tools, strong emphasis	Click **Apply** or **OK**.
→	Series of menu selections	Select **File → Save**
`Monospace`	Command-line and configuration options, file names and directories, etc.	Refer to `readme.txt`
Italics	Reference to other documents or products, URLs, emphasis	See *Tectia Client User Manual*
`Monospace Italics`	Replaceable text or values	rename *oldfile newfile*
#	In front of a command, # indicates that the command is run as a privileged user (root).	`# rpm --install package.rpm`
$	In front of a command, $ indicates that the command is run as a non-privileged user.	`$ sshg3 user@host`
\	At the end of a line in a command, \ indicates that the command continues on the next line, but there was not space enough to show it on one line.	`$ ssh-keygen-g3 -t rsa \` `  -F -c mykey`

🛈 Note

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

⚠ Caution

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

1.4.1 Operating System Names

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

- **Unix** refers to the following supported operating systems:

- HP-UX

- IBM AIX

- Red Hat Linux, SUSE Linux

- Linux on IBM System z

- 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.

1.5 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 http://answers.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/.

1.6 Terminology

The following terms are used throughout the Tectia Server for IBM z/OS documentation.

client computer
    The computer from which the Secure Shell connection is initiated.

Connection Broker
    The Connection Broker is a component included in the Tectia Server for IBM z/OS client tools. It consists of the **ssh-broker-g3** process and the **ssh-broker-ctl** control process. The Connection Broker handles all cryptographic operations and authentication-related tasks.

FTP-SFTP conversion
    Tectia SOCKS Proxy can automatically capture FTP connections on the client and convert them to SFTP and direct them to an SFTP server running Tectia Server, or another vendor's Secure Shell server software.

host key pair

> A public-key pair used to identify a Secure Shell server. The private key file is accessible only to the server. The public key file is distributed to users connecting to the server.

remote host

> Refers to the other party of the connection, client computer or server computer, depending on the viewpoint.

scpg3

> A command-line Secure Shell client for secure copy. **scpg3** is a secure replacement for remote copy (**rcp**) and provides easy secure non-interactive file transfers.

Secure Shell client

> A client-side application that uses the Secure Shell version 2 protocol, for example **scpg3**, **sftpg3**, or **sshg3** of Tectia client tools for z/OS.

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.

sftpg3

> A command-line Secure Shell client for secure file transfer. **sftpg3** is a secure replacement for FTP and provides a user interface for interactive file transfers and a batch mode for unattended file transfers.

SOCKS Proxy

> Tectia SOCKS Proxy is used for transparent FTP tunneling and FTP-SFTP conversion. It consists of the **ssh-socks-proxy** process and **ssh-socks-proxy-ctl** control process.

sshd2

> The main Secure Shell server daemon of Tectia Server for IBM z/OS.

sshg3

> A command-line Secure Shell client that can be used as a secure replacement for Telnet and other unsecured terminal applications. **sshg3** can also be used for remote command and job execution, and for creating secure tunnels for TCP applications.

Tectia client tools for z/OS

> Tectia Server for IBM z/OS includes client tools, which consist of the Connection Broker, Secure Shell command-line tools (sshg3, scpg3, sftpg3), auxiliary command-line tools, and the SOCKS Proxy .

Tectia client/server solution

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

Tectia Server for IBM z/OS

Tectia Server for IBM z/OS is a server-side component where Secure Shell clients connect to. It consists of two processes: **sshd2** and **ssh-certd** (the Certificate Validator). Tectia Server for IBM z/OS provides normal Secure Shell connections and supports secure TN3270 connectivity, transparent FTP tunneling, FTP-SFTP conversion and enhanced file transfer features on IBM z/OS.

Tectia SSH Assistant

The Tectia SSH Assistant ISPF application provides an interface for installing, configuring and running Tectia Server for IBM z/OS using traditional MVS tools (ISPF and JCL), without requiring the use of the Unix shell.

transparent FTP tunneling

An FTP connection transparently encrypted and secured by a Secure Shell tunnel.

tunneled application

A TCP application secured by a Secure Shell connection.

user key pair

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

Chapter 2 Installing Tectia Server for IBM z/OS

This chapter contains instructions on installing Tectia Server for IBM z/OS.

2.1 Preparing for Installation

This section lists the necessary prerequisites for installing Tectia Server for IBM z/OS.

2.1.1 System Requirements

The following operating system versions are supported as Tectia Server for IBM z/OS platforms:

- z/OS 2.2, 2.3 and 2.4

The following *minimum* hardware is required:

- IBM System z10 (or later)

- 1 GB RAM for hundreds of simultaneous tunnels

- 200 MB free disk space, 200 cylinders (includes client components)

  During installation, an additional 300 MB of temporary disk space is required.

- TCP/IP connection

2.1.2 Permission Requirements

Before Installation

Before you start installing Tectia Server for IBM z/OS, make sure the following requirements are met:

File system requirements

  Write access to the `/opt` directory is required during the installation.

User account requirements for installing the server

- The user account installing the product must have an OMVS segment, UID 0 and RACF SPECIAL privilege.

User account requirements for running the server

- The user account running the server must have an OMVS segment and the UID 0.

- If the `BPX.DAEMON FACILITY` class profile is defined, the user must have read access to it.

Requirements for user accounts to support access via Tectia Server

- *Required:* An OMVS segment

  Users who are to have access to in-bound SFTP or SSH require an OMVS segment defined in their profile. If a shell program is specified, it must be the pathname of an executable z/OS UNIX shell program; if omitted, the default shell program defined in z/OS UNIX customization is used.

- *Optional:* A home directory. It is required if public key user authentication is used or if the account requires user-specific configuration, for example, environment variables for the file transfer subsystem.

Requirements for user accounts that run Tectia client programs

- *Required:* An OMVS segment

- *Optional:* A home directory. It is required if public key user authentication is used or if the account requires user-specific configuration, for example, profiles for remote hosts.

Library requirements

- The Tectia SSH Assistant application requires the Rexx runtime or Rexx alternate libraries to execute. The Rexx Alternate Library SEAGALT (for example, `FAN140.SEAGALT` or `IBM.REXX.SEAGALT`, etc.), which is shipped as part of z/OS since version 1.9, may be used to satisfy this requirement. SEAGALT must be available in the linklist or in a STEPLIB allocated to your TSO session.

- *Optional:* By default SSZASST requires SDSF to operate. This dependency to SDSF can be removed in the Tectia SSH Assistant's Installation settings and defaults (0 SETM) panel, using option 0 (0.0) and changing the setting. TSO console ===> Y.

Permissions for storing keys in SAF

If the server host key or the user keys are going to be stored in the System Authorization Facility (SAF), additional permissions are required. See Section 5.2 for more information.

During Installation

The following additional requirements will be handled during the installation with Tectia SSH Assistant:

- The Tectia SSH Assistant ISPF application uses the **extattr** command to make the server program, `/opt/tectia/sbin/sshd2`, program-controlled. To issue the command, the user account running the setup must have read access to the `BPX.FILEATTR.PROGCTL` facility. These requirements will be fulfilled during the installation by running the job generated by the Tectia SSH Assistant option **1.1 INSTUSER**.

- It is recommended that a user account, `SSHD2`, is created for running Tectia Server for IBM z/OS. This will be done during the installation by running the job generated by the Tectia SSH Assistant option **1.3 ADDSSHDU**.

- `CEE.SCEERUN` and `CEE.SCEERUN2` libraries must be available in `LPALIB` or `LNKLST`, and `CEE.SCEERUN2` must be program-controlled. These requirements will be fulfilled during the installation by running the job generated by the Tectia SSH Assistant option **1.2 CPGMCTL**.

- The server must be allowed to listen to port 22 (or other configured Secure Shell port). The access will be granted during the installation by running the job generated by the Tectia SSH Assistant option **1.6 SERVAUTH**.

2.1.3 System Limits and Requirements

Check the definitions in BPXPRMxx. The following guidelines are approximate and apply to both the server and the clients.

Tectia requires an address space size of at least 200 MB (MAXASSIZE).

Each concurrent connection typically needs 2 to 3 processes (MAXPROCSYS, MAXPROCUSER). The user limit may require adjusting if many concurrent connections are run under one user ID.

For each concurrent connection Tectia requires:

- file descriptors (MAXFILEPROC)

- IPv4 sockets (MAXSOCKET for file system type INET)

- IPv6 sockets (MAXSOCKET for file system type INET6)

- AF_UNIX sockets (MAXSOCKET for file system type UDS)

The limits suggested in the z/OS UNIX System Services Planning book are typically adequate except maybe for AF_UNIX. Tectia uses 3 to 4 AF_UNIX sockets for each concurrent connection.

Note that Tectia is not defined to z/OS Workload Manager (WLM) and runs the same goals as UNIX System Services (USS). If you feel the need to make changes to USS goals, please consult the IBM Redbook *System Programmer's Guide to: Workload Manager*, Chapter *UNIX System Services considerations*.

2.1.4 Directories and Data Sets

USS

The directory structure under Unix System Services (USS) is shown below. The space requirements are approximate upper limits.

`/opt/tectia` symlink

> Symbolic link to the zFS (Space: 200 MB, 200 Cyls, read/write) that contains executable binaries, setup scripts, configuration files, server key files, manual pages, documentation, license agreement, example JCL scripts.
>
> After installation, Tectia Server for IBM z/OS consists of a directory structure under the zFS file system defined for it. The zFS is mounted onto a mount point defined at installation, and pointed to by the `/opt/tectia` symlink.

`$HOME/.ssh2`

> Each z/OS user account that is accessed via Tectia Server or that runs SSH client programs normally has a USS home directory (`$HOME`, for example, `/u/home1/username`) and under it a `.ssh2` subdirectory. The `.ssh2` directory contains the user's configuration files and keys. The home directory is required if public key user authentication is used and if user-specific configuration is needed.
>
> Space: 128 kB, read/write
>
> The runtime programs of Tectia Server for IBM z/OS create the `$HOME/.ssh2` directories as needed. If you want these directories to be a link to some other spot in your directory hierarchy, create the link before running the program.

`/tmp`

> Contains server process ID files and the default `STDOUT` and `STDERR`.
>
> Space: 256 kB (300 MB during installation), read/write/sticky.
>
> The `/tmp` directory must exist in advance and it must be user-writable and have the sticky bit on.
>
> The sticky attribute for `/tmp` can be checked by observing the output of **ls** and verifying that the letter `t` is present in the permissions field:

```
:> ls -ld /tmp/.
drwxrwxrwt  89 WEBSRV   SYS1       49152 Apr 11 14:43 /tmp/.
```

> If the permissions differ from `rwxrwxrwxt`, they must be adjusted with **chmod**:

```
:> chmod 1777 /tmp/.
```

`/tmp/ssh-`*`username`*

> Contains users' temporary files used in Secure Shell agent forwarding.

> The agent forwarding status files are temporary and valid only while the actual user process is running.

> Space: 12 kB for each user, read/write

MVS

Although this version of Tectia Server for IBM z/OS must be installed in a USS file system and use the directory structure shown above, the server supports the transfer of MVS files and all the programs can be executed in JCL by `BPXBATCH`, `BPXBATSL`, and `oshell`.

2.1.5 Licensing

Tectia Server and client tools for z/OS require valid licenses, which are provided separately.

> ### ℹ Note
>
> The installation packages do not contain license files. The license files must be requested separately from SSH technical support.

For instructions on uploading the licenses to your z/OS host, see Section 2.1.6.

2.1.6 Uploading Files Required for Installation

> ### ℹ Note
>
> If you have an existing installation of Tectia Server for IBM z/OS version 6.x, you can skip to Section 2.2.

The following files need to be uploaded (you can find detailed instructions below) in binary mode to your z/OS host:

- *Tectia SSH Assistant XMIT file* `SSZASST.V060611.B`*`XXXX`*`.XMIT`

  Extract the XMIT file from `sszasst-6.6.11.`*`XXX`*`.zip` to a temporary location on your local machine.

  (Replace the '*x*'s in the file names with the correct build number.)

  Upload the file with the following data set attributes:

  Data set organization: `RECFM=FB`
  Record length: `LRECL=80`
  Space allocation unit: `TRACKS`
  Primary space allocation: `PRIMARY=15`

- *Tectia Server for IBM z/OS product tar archive* `ssh-tectia-server-zos-6.6.11.`*XXX*`-ibmzos.tar.Z`

  Extract the product tar archive from `tectia-server-zos-6.6.11.`*XXX*`-ibmzos-comm.tar` to a temporary location on your local machine.

  The product tar archive `ssh-tectia-server-zos-6.6.11.`*XXX*`-ibmzos.tar.Z` can be transferred to either the Unix file system or to a sequential data set. Approximately 55M or 120 cylinders are required.

  You can use the following data set attributes:

  Data set organization: `RECFM=U`
  Maximum block size: `BLKSIZE=32256`
  Record length: `LRECL=0`
  Space allocation unit: `CYLINDERS`
  Primary space allocation: `PRIMARY=120`

- *Product licenses tar archive*

Check if you have some Secure Shell software, for example earlier versions of Tectia products or IBM Ported Tool for z/OS (OpenSSH), running on the machine where you are planning to install the new Tectia version.

In the following sections we provide you with the required commands for uploading the files using OpenSSH, FTP or FTP via cURL.

Uploading Installation Files Using OpenSSH

If an OpenSSH server is already running on the host, use **sftp** to transfer the installation files in binary format to the USS file hierarchy, from where the XMIT file can be copied to a data set with the **cp** command:

```
cp -P 'recfm=fb,lrecl=80,blksize=6160,space=(trk,(15,15))' \
SSZASST.V060611.BXXXX.XMIT //SSZASST.V060611.BXXXX.XMIT
```

(Replace the *xxxx* in the XMIT file name with the correct build number.)

Before installing Tectia Server 6.6 for IBM z/OS, stop any OpenSSH servers running on port 22, or change their listener port. You do not need to uninstall the OpenSSH software.

Proceed with the installation normally as described in .

Uploading Installation Files Using FTP

Connect to your z/OS host and change file transfer mode to binary:

```
ftp USER@zoshost
ftp> binary
```

Upload the Tectia SSH Assistant XMIT file:

```
ftp> quote site RECFM=FB LRECL=80 TRACKS PRIMARY=15
ftp> put SSZASST.V060611.BXXXX.XMIT
```

(Replace the *xxxx* in the XMIT file name with the correct build number.)

Upload the product tar archive (extracted from `tectia-server-zos-6.6.11.XXX-ibmzos-comm.tar`):

```
ftp> quote site RECFM=U BLKSIZE=32256 LRECL=0 CYLINDERS PRIMARY=120
ftp> put ssh-tectia-server-zos-6.6.11.xxx-ibmzos.tar.Z SSZ.V6611xxx.TARZ
```

(Replace the *xxx* in the tar archive and destination data set names with the correct build number.)

Upload the licenses tar archive to the user's home directory:

```
ftp> cd /u/user/
ftp> put licences.tar
ftp> quit
```

Uploading Installation Files Using FTP via cURL

Upload the Tectia SSH Assistant XMIT file:

```
curl -v -u USER -Q 'site RECFM=FB LRECL=80 TRACKS PRIMARY=15' \
-T SSZASST.V060611.BXXXX.XMIT ftp://zoshost
```

(Replace the *xxxx* in the XMIT file name with the correct build number.)

Upload the product tar archive (extracted from `tectia-server-zos-6.6.11.XXX-ibmzos-comm.tar`):

```
curl -v -u USER -Q 'site RECFM=U BLKSIZE=32256 LRECL=0 CYLINDERS PRIMARY=120' \
-T ssh-tectia-server-zos-6.6.11.xxx-ibmzos.tar.Z \
ftp://zoshost/SSZ.V6611xxx.TARZ
```

(Replace the *xxx* in the tar archive and destination data set names with the correct build number.)

Upload the licenses tar archive to the user's home directory:

```
curl -v -u USER -T licences.tar ftp://zoshost//u/user
```

2.2 Upgrading from Tectia Server for IBM z/OS Version 6.x

Note

> Before starting the upgrade, take backups of any previous installation binaries you may want to retain for possible future use.

2.2.1 Uploading Files Required for Upgrade

Upload the *Tectia SSH Assistant XMIT file* (`SSZASST.V060611.BXXXX.XMIT`), the *Tectia Server for IBM z/OS product tar archive* (`ssh-tectia-server-zos-6.6.11.XXX-ibmzos_1_13.tar.Z`), and the *licenses tar archive* to your z/OS system.

SSZASST.V060611.B*XXXX*.XMIT is included in `sszasst-6.6.11.XXX.zip` and the product tar archive is included in the `tectia-server-zos-6.6.11.XXX`-ibmzos_1_13-comm.zip archive.

You can transfer the files directly to DASD using FTADV or site commands to set the data set attributes. In the following we provide you with example commands for uploading the files using Tectia and OpenSSH clients.

Using Tectia Client

```
sftpg3 USER@zoshost ❶
sftp> binary ❷
sftp> sput SSZASST.V060611.BXXXX.XMIT /ftadv:LR=80,REC=FB,SU=TRKS,PRI=15,T=PS,S=NO,\
F=STREAM/__SSZASST.V060611.BXXXX.XMIT ❸
sftp> sput ssh-tectia-server-zos-6.6.11.XXX-ibmzos_1_13.tar.Z /ftadv:RECFM=U,\
BLKSIZE=32256,LRECL=0,CYLINDERS,PRI=120,T=PS,S=NO/__SSZ.V662XXX.TARZ ❹
sftp> sput licenses-6.6.tar ❺
```

❶   Connect to *zoshost*.

❷   Set file transfer mode to binary.

❸   Upload the XMIT file using FTADV to set the data set attributes. (Replace the *xxxx* in the XMIT file name with the correct build number.)

❹   Upload the product tar archive using FTADV to set the data set attributes. (Replace the *xxx* in the tar archive and destination data set names with the correct build number.)

❺   Upload the licenses tar archive.

Using OpenSSH Client

```
sftp USER@zoshost
sftp> binary
sftp> put SSZASST.V060611.BXXXX.XMIT /ftadv:LR=80,REC=FB,SU=TRKS,PRI=15,T=PS,S=NO,\
F=STREAM/__SSZASST.V060611.BXXXX.XMIT
sftp> put ssh-tectia-server-zos-6.6.11.XXX-ibmzos_1_13.tar.Z /ftadv:RECFM=U,\
BLKSIZE=32256,LRECL=0,CYLINDERS,PRI=120,T=PS,S=NO/__SSZ.V662XXX.TARZ
sftp> put licenses-6.6.tar
```

2.2.2 Upgrading to Tectia Server for IBM z/OS 6.6.11

Preparing for the Product Upgrade

After you have uploaded the files required for the upgrade, do the following on your z/OS host:

1.   Receive the Tectia SSH Assistant application data set (replace the *xxxx* in the name with the correct build number):

```
TSO RECEIVE INDSN(SSZASST.V060611.BXXXX.XMIT)
```

Press **Enter** repeatedly to page through the command output and when prompted, press **R** to replace the existing data set.

You should see this message in the end of the command output: `Restore successful to dataset '`*`prefix`*`.SSZASST.PDS'`.

2. Set up the application libraries by EXECing the Rexx script `$RECEIVE` that is inside the restored data set:

```
TSO EXEC 'prefix.SSZASST.PDS($RECEIVE)'
```

You should see this message in the end of the command output: `Restore successful to dataset '`*`prefix`*`.SSZASST.SKEL'`.

3. EXEC the Tectia SSH Assistant application:

```
TSO EXEC 'prefix.SSZASST.CEXEC(SSZ)'
```

4. Update the following settings in the Tectia SSH Assistant ISPF application:

   - **0.1 SETI**: `Version` and `Tarball`. Make sure you enter the correct `Version`. It must correspond with the untarred directory name of the product version you are upgrading to.

     ### (i) Tip

     > You can check the current `Version` by pressing `PF1 (help)`.

   - **0.2 SETO**: `DSName` and `Mountpoint`

     ### (i) Note

     > You only have to type in the `Version`. For the other fields (`Tarball`, `DSName` and `Mountpoint`) it is enough to blank the field out and press **Enter**. A suitable value based on the `Version` will automatically be added to the field.

   For more information on all the available settings, see Section 2.3.2.

5. Generate the required installation jobs. When updating an existing working installation, it is enough to successfully run jobs 7 to 13. When new licenses are to be installed, also job 14 must be run. To generate all the installation jobs at once, use the **1.99 GENALL** option.

   For descriptions of the installation jobs, see Section 2.3.3.

   ### (i) Note

   > If your installation has CEX cards, and you are updating from version 6.5.x or older, you must also run job 5.

Completing the Product Upgrade

The installing user must next do the following to complete the upgrade:

1. In Tectia SSH Assistant, go to **0 SETM** → **3 SETL** and define the installation log data set from which to extract the installation settings:

   ```
   Logfile   ===> '<prefix>.SSZ.INSTALL.LOG'
   ```

2. In **0.1 SETI** and **0.2 SETO**, check that the installation input and output settings are correct.

3. In **2.1 JOBS**, submit the following installation jobs:

   - `07 (SAVE)` - Save key data from previous installation (generated by action **1.7 SAVE**).

     > ℹ️ **Note**
     >
     > If the Certificate server is running, running this job may result in error code `0256`. You can ignore the error.

   - `08 (ZFSA)` - Define the installation zFS: allocate a new zFS data set, format and mount it onto the defined mount point, using names and attributes given in the settings panel **0.2 SETO** (generated by action **1.8 ZFS**).

   - `09 (LOAD)` - Load the installation zFS (generated by action **1.9 LOAD**).

   - `10 (REST)` - Restore key data from previous installation (generated by action **1.10 RESTORE**).

   - `11 (OPTT)` - Create the `/opt/tectia` symlink (generated by action **1.11 SYMLINK**).

     > ℹ️ **Note**
     >
     > This job will delete a previously existing `/opt/tectia` symlink. However, if the existing `/opt/tectia` is a directory and not a symlink, this job will fail. This is to avoid the accidental deletion of directory contents that may not have been backed up.
     >
     > To delete the `/opt/tectia` directory, use the `-r` option with the **rm** command. Note that to delete the directory, you must first unmount the existing mount point.

   - `12 (LIBS)` - Create PDS libraries for sample jobs and parameters (generated by action **1.12 SSZLIBS**).

   - `13 (PROC)` - Set up started task procedures (generated by action **1.13 PROCLIB**).

   - `14 (LICN)` - Install new licenses (generated by action **1.14 LICENCE**).

> **ℹ** **Note**
>
> If the installing user is not allowed to run the server, re-grant them console rights by running job `01 (IUSR)` (generated by action **1.1 INSTUSER**) again. The installing user must then log on again to acquire the new console rights.

4.   To restart the server, go to **4 TASK** → **1 TSRV** and enter **3 TSRVR**.

     To check the version of the running server, enter **5 TSRVQV**.

5.   *(Optional)* You can use job `U01ZDEL`, accessible via the Tectia SSH Assistant main menu option `5 UTIL`, to delete the previous installation ZFS. The job unmounts the installation ZFS, deletes the mount point, and deletes the installation ZFS data set (i.e., everything under `/opt/tectia`). Only the PROCs and samples will be retained.

> **⚠** **Caution**
>
> Job `U01ZDEL` uses the previous version number, so if you run "Generate installation jobs" (`1 GENJ`) multiple times for the same release, the job will remove the ZFS of the current installation.

2.3 Installing Tectia Server for IBM z/OS

Installing Tectia Server for IBM z/OS consists of the following steps:

1.   Uploading the files required for the installation to the z/OS host. See Section 2.1.6.

2.   Installing the Tectia SSH Assistant ISPF application. See Section 2.3.1.

3.   Defining the required settings for the installation. See Section 2.3.2.

4.   Generating the product installation jobs. See Section 2.3.3.

5.   *As the installing user:* submitting the installation jobs. See Section 2.3.4.

2.3.1 Installing the Tectia SSH Assistant ISPF Application

The Tectia SSH Assistant (**SSZASST**) ISPF application provides an interface for installing and configuring Tectia Server for IBM z/OS and its client tools. It is designed to simplify the process of installing the product tar archive appropriately and performing the multiple configuration tasks required using traditional MVS tools (ISPF and JCL), without requiring the use of the Unix shell.

1. If you have not yet done so, transfer the Tectia SSH Assistant application XMIT file and the Tectia
    Server for IBM z/OS product tar archive in binary mode to your z/OS system. For instructions, see Sec-
    tion 2.1.6.

2. On the z/OS host, receive the Tectia SSH Assistant data set via the following command (replace the
    xxxx in the XMIT file name with the correct build number):

    ```
    TSO RECEIVE INDSN(SSZASST.V060611.Bxxxx.XMIT)
    ```

    In response to the RECEIVE prompt, you may enter the usual parameters to control the creation of the
    received data set, or just press enter to take the defaults and create a data set called *prefix*.SSZASST.PDS.

3. Inside the restored data set you will find a Rexx script called $RECEIVE. EXEC the script to set up the
    application libraries:

    ```
    TSO EXEC 'prefix.SSZASST.PDS($RECEIVE)'
    ```

    (Alternatively, you can simply type EXEC next to $RECEIVE in a member list.)

    This Rexx will prompt for the HLQ under which the application libraries are to be set up, as well as op-
    tional VOLSER, if needed.

4. Press **Enter** repeatedly to page through the command output.

    The following libraries will be created, assuming default names:

    ```
    prefix.SSZASST.CEXEC
    prefix.SSZASST.ISPMLIB
    prefix.SSZASST.ISPPLIB
    prefix.SSZASST.ISPSLIB
    prefix.SSZASST.SKEL
    ```

5. The Tectia SSH Assistant application requires the Rexx runtime or Rexx alternate libraries to execute.
    The Rexx Alternate Library SEAGALT (for example, FAN140.SEAGALT or IBM.REXX.SEAGALT, etc.),
    which is shipped as part of z/OS since version 1.9, may be used to satisfy this requirement. Make sure
    that SEAGALT is available in the linklist or in a STEPLIB allocated to your TSO session.

    The following message indicates that a suitable Rexx runtime was not found:

    ```
    IRX0159E The run time processor EAGRTPRC is not available
    ```

    To solve the issue, add a line to the appropriate PARMLIB(PROGxx) member such as:

    ```
    LNKLST ADD NAME(LNKLST00) DSN(FAN140.SEAGALT) VOLUME(&SYSR1)
    ```

6. Set up the Tectia SSH Assistant application to be invoked. The simplest way to do this is to EXEC
    prefix.SSZASST.CEXEC(SSZ) directly, which will use LIBDEF to allocate the panel and skeleton lib-
    raries, assuming they share the same qualifiers as the Rexx library:

```
TSO EXEC 'prefix.SSZASST.CEXEC(SSZ)'
```

Alternatively, you can concatenate the libraries to the appropriate DDs in your TSO logon procedure, or copy their contents to allocated user ISPF data sets.

Figure 2.1. Tectia SSH Assistant main menu

The mode of operation of Tectia SSH Assistant follows a probably familiar approach of collecting settings, generating JCL jobs and configuration files, and then executing those jobs. Since there are many steps which must be run by a privileged user, such as granting RACF permissions, defining file systems, etc., the install jobs may be run by other users than the one who generated them.

Table 2.1. ISPF Tectia SSH Assistant Menu Structure

Menu item	Description
0 SETM	Installation settings and defaults submenu
0.1 SETI	Define settings for installation input
0.2 SETO	Define settings for installation output
0.3 SETL	Load settings profile from logged definition
1 GENJ	Generate installation jobs
1.1 INSTUSER	Grant permissions to user doing install
1.2 CPGMCTL	Ensure C library program-controlled
1.3 ADDSSHDU	Set up SSH Server user
1.4 ADDSOXPU	Set up SOCKS Proxy Server user
1.5 CSFSERV	ICSF permissions
1.6 SERVAUTH	Port 22 control
1.7 SAVE	(Save previous installation key data)
1.8 ZFS	Define installation ZFS
1.9 LOAD	Load installation ZFS
1.10 RESTORE	(Restore previous installation key data)
1.11 SYMLINK	Create /opt/tectia symlink
1.12 SSZLIBS	Sample JCL and PARM libraries
1.13 PROCLIB	Set up started task procedures
1.14 LICENCE	Install licenses from supplied tarball
1.15 KEYGEN	Generate server host keys
1.99 GENALL	Generate all jobs
2 INST	Perform the step-by-step installation
2.1 JOBS	Member list of generated installation jobs (*prefix*.SSZ.INSTALL.CNTL)
2.2 LOG	Browse log of settings and executed jobs (*prefix*.SSZ.INSTALL.LOG)
3 CONF	Manage configuration files
3.1 ETCD	View the installation etc directory
3.2 SSHD2	SSHD2 server configuration file (/opt/tectia/etc/sshd2_config)
3.3 CERT	Certificate Validator configuration file (/opt/tectia/etc/ssh_certd_config)
3.4 SOXP	SOCKS Proxy configuration file (/opt/tectia/etc/ssh-socks-proxy-config.xml)
4 TASK	Start/stop/modify started tasks
4.1 TSRV	Control the SSH server
4.2 TCRT	Control the certificate server
4.3 TSXP	Control the Socks proxy server
5 UTIL	Utility jobs to manage the installation

2.3.2 Installation Settings and Defaults

Set up the settings for the installation in the Tectia SSH Assistant's **Installation settings and defaults (0 SETM)** panel. The settings for installation input (0.1 SETI) and output (0.2 SETO) are described in more detail in the following, and you can also consult the online help.

The settings will be saved in the installation log file, as well as in the user's application profile. The facility for another user to load these settings from the log file is provided by the **Load settings profile from logged definition (0.3 SETL)** option.

0.1 SETI - Settings for installation input


```
SSZ                 Define Settings and Defaults          2016/05/09 11:26
Command ===>

Enter the input settings to use for customizing SSH install jobs.

HLQ        ===> SSZ_____    Installation high-level qualifier
Version    ===> 6.6.0.9___    SSZ Version number
Installer  ===> BERT____      UID 0 user who will perform installation
Tarball    ===> 'FRED.SSZ.V6609.TARZ'_____  >
Licences   ===> /tmp/licenses-6.6.tar_____  >

Installer tailored batch jobs output dataset details:
DSName     ===> 'FRED.SSZ.INSTALL.CNTL'_____
Logfile    ===> 'FRED.SSZ.INSTALL.LOG'_____
Volser     ===> _____   Unit ===> _____    Storclas ===> _____

Default jobcards for install jobs:
 //_____JOB_,,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),_____
 //_____NOTIFY=&SYSUID_____
 _____
 _____
```

Figure 2.2. Tectia SSH Assistant Settings for installation input (0.1 SETI)

The following table lists the settings in the **Define settings for installation input (0.1 SETI)** panel used for customizing the install jobs.

Table 2.2. Define settings for installation input (0.1 SETI)

Setting	Description	Example value
HLQ	Data set high-level qualifier for this install. The high-level qualifier will be used to name data sets, such as PARMLIB and SAMPLIB, created during installation.	SSZ
Version	Version number of the SSH package to install. The version number will usually be part of the name of the original product install tar archive, which will contain a directory named ./ssh-tectia-server-zos-6.6.11.*XXX*. *Make sure you enter the correct version number!* It must correspond with the untarred directory name.	6.6.11.123
Installer	ID of the UID=0 user who will run the installation. A user with the needed authority is required to run jobs and perform other actions to complete the installation. Jobs will be generated to grant this user certain rights.	BERT
Tarball	Name of the TAR file from which to install Tectia Server for IBM z/OS. The tar archive may be a Unix file or a data set, distinguished by the presence or absence of a leading Unix path symbol. The upload of the tar archive should have been done in binary mode, and for a data set target, its attributes should be RECFM=0 BLKSIZE=32256 LRECL=0 CYLINDERS PRIMARY=120.	Unix file: /tmp/ssz-6.6.11.123.tar.Z Data set: 'FRED.SSZ.V6611100.TARZ'
Licenses	Name of the TAR file containing the product licenses. The tar archive may be a Unix file or a data set, distinguished by the presence or absence of a leading Unix path symbol. If the tar archive is gzipped, gzip must be available.	Unix file: /tmp/licenses-6.6.tar.gz, /u/fred/licenses-6.6.tar Data set: //'FRED.SSZ.LIC.TAR'
Installer tailored batch jobs output data set details		
DSName	Name of the tailored batch jobs output data set. If the DSN exists, it must be a partitioned data set. If it does not exist, it will be created.	'FRED.SSZ.INSTALL.CNTL'
Logfile	Name of the installation log data set. The DSN will not be qualified and will be created as a sequential file, deleting it if it already exists. The optional volser and unit will be used if supplied.	'FRED.SSZ.INSTALL.LOG'
Volser *(optional)*	Volume for tailored jobs output data set. A volser may optionally be supplied to be used in creating the output data set for installation batch jobs.	HDSYS1

Setting	Description	Example value
Unit *(optional)*	Unit for tailored jobs output data set. A unit may optionally be supplied to be used in creating the output data set for installation batch jobs.	SYSDA
Storclas *(optional)*	Storage class for tailored jobs output data set. A storage class may optionally be supplied to be used in creating the output data set for installation batch jobs.	VENDORDS
Default jobcards for install jobs		
	Your local Job Control Language (JCL) Job Cards. You can specify as many as four default job cards which will be used in generating configuration and sample jobs. The jobname may be left blank; it will be generated based on the installer user ID and the job name.	

0.2 SETO - Settings for installation output

Figure 2.3. Tectia SSH Assistant Settings for installation output (0.2 SETO)

The following table lists the settings in the **Define settings for installation output (0.2 SETO)** panel used for customizing the install jobs.

Table 2.3. Define settings for installation output (0.2 SETO)

Setting	Description	Example value
Target ZFS details:		
DSName	Name of the data set for the installation zFS. The DSN will be used to tailor jobs to allocate and define the zFS, as well as mount commands and BPXPRM parameters.	'SSZ.V66110*xxx*.ZFS'
Volser	Volume for the installation zFS. A volser may optionally be supplied to be used to tailor jobs to allocate and define the zFS. If you do not define Volser, you must define Storclas (below).	SSZ001
Unit	Unit for the installation zFS. A unit may optionally be supplied to be used to tailor jobs to allocate and define the zFS. If you do not define Unit, you must define Storclas (below).	SYSDA
Storclas	Storage class for the installation zFS. A storage class may optionally be supplied to be used to tailor jobs to allocate and define the zFS. If you defined Volser and Unit, leave this empty.	VENDORZF
Mountpoint	Mountpoint name for the installation zFS. The mountpoint name specifies the point in the Unix file system where the installation zFS will be mounted. It is used in generating jobs, commands and parameters.	/u/vendor/ssz-66110*xxx*
System libraries to receive procs and parms:		
Proclib	Name of the system library to receive procs. The proclib name is used in generating jobs to install cataloged procedures for started tasks for the SSH server and other server tasks.	USER.PROCLIB
Parmlib	Name of the system library to receive parms. The parmlib name is used in generating procedures for started tasks such as the SSH server.	USER.PARMLIB
Started task details:		
SSHD user	ID of the UID=0 user under which to run the server. A user with the needed authority is required to run the SSH server. The value supplied here will be used to create jobs to set up this user and grant the required rights.	SSHD2
SSHD proc	SSH server cataloged procedure name. The name of the started task procedure to run the SSH server supplied here will be used in tailoring and loading the procedure to the proclib.	SSHD2
CERT proc	Certificate Validator cataloged procedure name. The name of the started task procedure to run the Certificate server supplied here will be used in tailoring and loading the procedure to the proclib.	SSHCERTD
SOXP user	ID of the user under which to run the SOCKS Proxy. A user with the needed authority is required to run the SOCKS Proxy. The value supplied here will be used to generate jobs to set up this user and grant the required rights.	SSHSP

Setting	Description	Example value
SOXP proc	SOCKS Proxy cataloged procedure name. The name of the started task procedure to run the SOCKS server supplied here will be used in tailoring and loading the procedure to the proclib.	

0.3 `SETL` - SSH Settings Log

Use this panel to indicate the SSH settings log to use.

SSH settings defined by another user may be loaded from the log file where they have been saved, enabling another user to continue the installation using those settings.

Figure 2.4. Tectia SSH Assistant Specify Settings Log (0.3 SETL)

2.3.3 Generating Product Installation Jobs

```
SSZ                    Generate Installation Jobs            2016/05/09 11:37
Option ===>

  1 INSTUSR              Grant permissions to user doing install
  2 CPGMCTL              Ensure C library program-controlled
  3 ADDSSHDU             Setup SSH Server user
  4 ADDSOXPU             Setup Socks Proxy Server user

  5 CSFSERV              ICSF permissions
  6 SERVAUTH             Port 22 control

  7 SAVE                 (Save previous installation key data)
  8 ZFS                  Define installation ZFS
  9 LOAD                 Load installation ZFS
 10 RESTORE              (Restore previous installation key data)

 11 SYMLINK              Create /opt/tectia symlink
 12 SSZLIBS              Sample JCL and PARM libraries
 13 PROCLIB              Set up started task procs
 14 LICENCE              Install licences from supplied tarball
 15 KEYGEN               Generate server host keys

 99 GENALL               Generate all jobs  ( Edit: N )
```

Figure 2.5. Tectia SSH Assistant Generate Installation Jobs submenu (1 GENJ)

Tailoring of the required installation jobs is performed from the **Generate installation jobs (1 GENJ)** submenu.

> ## Note
>
> The exact jobs required depend on the type of installation and a knowledgeable user may skip those not required. For instance, in the case of updating an existing working installation, it is enough to successfully run jobs 7 to 13 (and job 14 when new licenses are to be installed) - see the section called "Completing the Product Upgrade".

This step merely generates the jobs; nothing is run at this stage. You should inspect the generated jobs carefully, as well as verify them with TYPRUN=SCAN on the generated job cards. The JCLs must be run by the authorized installing user to have effect (see Section 2.3.4.

Selecting any of the options in this panel will result in the presentation of a file-tailored JCL job to perform the task in question. You can also use the **99 GENALL** option to generate all the jobs at once.

The options are listed in Table 2.4, followed by a more detailed description of each of them.

Table 2.4. Generate Installation Jobs submenu (1 GENJ)

Option	Description	The generated job must be run
1.1 INSTUSER	Grant permissions to user doing install	On first install
1.2 CPGMCTL	Ensure C library program-controlled	On first install
1.3 ADDSSHDU	Set up SSH Server user	On first install
1.4 ADDSOXPU	Set up SOCKS Proxy Server user	If the SOCKS Proxy is to be run as a started task
1.5 CSFSERV	ICSF permissions	If access to ICSF is to be restricted
1.6 SERVAUTH	Port 22 control	If access to the SSH port is to be controlled
1.7 SAVE	Save previous installation key data	On each upgrade (to carry customizations forward)
1.8 ZFS	Define installation ZFS	On each install/upgrade
1.9 LOAD	Load installation ZFS	On each install/upgrade
1.10 RESTORE	Restore previous installation key data	If the complementary SAVE job has been run
1.11 SYMLINK	Create /opt/tectia symlink	On each install/upgrade
1.12 SSZLIBS	Sample JCL and PARM libraries	On each install/upgrade
1.13 PROCLIB	Set up started task procedures	On each install/upgrade
1.14 LICENCE	Install licenses from supplied tarball	On first install and whenever licenses change
1.15 KEYGEN	Generate server host keys	On first install
1.99 GENALL	Generate all jobs	(On first install)

1.1 `INSTUSER` - Grant Permissions to User Doing Install

The user who is going to install, configure and run Tectia Server for IBM z/OS requires authority to make changes normally restricted to special users. You may use any user with RACF SPECIAL and UID 0 to perform this role.

In addition, the installing user needs some authorities which may well not be granted already. This step generates a job which will allow the installing user to permit program-control rights and to issue console commands. The job must be run at least on the first install of the product.

1.2 `CPGMCTL` - Ensure C Library is Program-controlled

It is necessary for the C Runtime Library to be marked program-controlled in order for Tectia Server for IBM z/OS to perform functions such as allowing authenticated users to log on.

This step generates a job which will allow the installing user to alter the C Runtime Library program-control status. The job must be run at least on the first install of the product.

1.3 `ADDSSHDU` - Set Up SSH Server User

Tectia Server for IBM z/OS should be run by a user dedicated for that purpose and granted the appropriate rights.

This step generates a job which will allow the installing user to define the SSH server started-task user, granting it permissions to run as a daemon and to create SMF records, as well as ownership of the SSH server and Certificate server started tasks. The job must be run at least on the first install of the product.

1.4 `ADDSOXPU` - Set Up SOCKS Proxy Server User

The SOCKS Proxy server should be run by a user dedicated for that purpose and granted the appropriate rights.

This step generates a job which will allow the installing user to define the SOCKS Proxy server started-task user. The started-task user will own the SOCKS Proxy server started task and have the right to change its job name. The home directory required by the started-task user will also be created.

If the SOCKS Proxy is to be run as a started task, this job must be run at least on the first install of the product.

1.5 `CSFSERV` - ICSF Permissions

To take advantage of cryptographic hardware and to manage its use, we recommend you to define access permissions.

This step generates a job which will grant permissions to all users to access certain features of ICSF cryptographic support needed for efficient SSH operations. If access to ICSF is to be restricted, this job should be run at least on the first install of the product.

1.6 `SERVAUTH` - Port 22 Control

Access to the SSH port may be controlled via the servauth RACF class, limiting binding this port to specific UIDs.

This step generates a job which will deny general access to the SSH port and allow only the defined SSH server user and the OpenSSH SSHDAEM user to bind the port. This job should be run if you want to control access to the SSH port (a recommended but not necessary practice).

1.7 `SAVE` - Save Previous Installation Key Data

When upgrading or re-installing, it is convenient to reuse some data from the previous installation.

This step generates a job which will create a tar archive of the contents of the customization and ephemeral data directories `etc` and `var` for later restoration after the install. This job should be run when upgrading an existing installation and you want to carry customizations forward.

1.8 `ZFS` - Define Installation ZFS

The Tectia Server for IBM z/OS installation is done into a separate zFS data set for control and manageability.

This step generates a job which will allocate a new zFS data set, format and mount it onto the defined mount point, using names and attributes given in the settings panel. This job must be run on each install or upgrade of the product.

1.9 `LOAD` - Load Installation ZFS

The Tectia Server for IBM z/OS product is supplied as a tar file which is loaded into a previously defined and mounted zFS.

This step generates a job which will load the product into the file system and ensure that the appropriate ownership, permissions and authorizations are set. This job must be run on each install or upgrade of the product.

1.10 `RESTORE` - Restore Previous Installation Key Data

When upgrading or re-installing, it is convenient to reuse some data from the previous installation.

This step generates a job which will restore the contents of the customization and ephemeral data directories `etc` and `var` saved prior to the install. This job should be run if its complementary SAVE job had been run.

1.11 `SYMLINK` - Create `/opt/tectia` Symlink

The Tectia Server for IBM z/OS product must be run from the path `/opt/tectia`, which is a symlink to the installation location.

This step generates a job which will create a symlink to the product installation directory. It is necessary for `/opt` to be mounted read-write when this job is run. This job must be run on each installation or upgrade.

1.12 `SSZLIBS` - Sample JCL and PARM Libraries

Sample jobs for a variety of SSH tasks as well as the recommended environment settings are provided.

This step generates a job which will create PDS libraries for the sample jobs and parameters, loading them with appropriately tailored content. This job should be run on each installation or upgrade.

1.13 `PROCLIB` - Set Up Started Task Procedures

Customized procedures for the Tectia Server for IBM z/OS started tasks are installed to the specified system proclib.

This step generates a job which will install tailored JCL procedures for the SSH server, Certificate server and SOCKS Proxy to the proclib requested in settings, as well as installing a BPXPRMSZ member in the requested

parmlib to facilitate mounting the SSZ zFS at OMVS start time. This job must be run on each installation or upgrade.

1.14 `LICENCE` - Install Licenses From Supplied Tarball

The Tectia Server and client tools for z/OS licenses are supplied in a separate tar file for installation.

This step generates a job which will extract the product licenses from the tar file supplied separately by SSH Customer Support, converting them and installing them in the required location. If the tar file is gzipped (`.gz`), the *gzip* utility must be available in the path. This job must be run on first install and whenever licenses change. On other occasions, it is easier to use the SAVE/RESTORE jobs to carry the licenses forward.

1.15 `KEYGEN` - Generate Server Host Keys

An SSH server public/private host key pair is required for server operations.

This step generates a job which will create a set of host keys (2048-bit RSA by default) for the SSH server and install them in the required location with correct permissions. This job must be run on first install. For a re-installation or upgrade, it may be desirable to reuse existing host keys via the SAVE/RESTORE jobs and skip this step.

2.3.4 Running the Product Installation Jobs

Once the product installation jobs have been generated as instructed in Section 2.3.3, the installing user must run them.

As the installing user, do the following:

1. EXEC the Tectia SSH Assistant application:

    ```
    TSO EXEC 'prefix.SSZASST.CEXEC(SSZ)'
    ```

2. In Tectia SSH Assistant, go to **0 SETM** → **3 SETL** and define the installation log data set from which to extract the installation settings:

    ```
    Logfile   ===> '<prefix>.SSZ.INSTALL.LOG'
    ```

3. In **0.1 SETI** and **0.2 SETO**, check that the installation input and output settings are correct.

4. The **Perform the step-by-step installation (2 INST)** submenu provides a member list (**1 JOBS**) of generated jobs in the order they should be run. Submit and check the installation jobs one by one in the order they are listed in. You should see a list of jobs that correspond to the actions in Section 2.3.3. Each job run updates the installation log with its results. After successful completion, the SSH server and clients are installed, with all necessary permissions set up.

```
MEMBER LIST   FRED.SSZ.INSTALL.CNTL                    Row 0000001 of 0000015
Command ===>                                           Scroll ===> PAGE
         Name      Prompt       Size   Created       Changed          ID
........ X01IUSR                46     2016/05/09    2016/05/09 08:02:36  FRED
........ X02CPCT                21     2016/05/09    2016/05/09 08:02:37  FRED
........ X03SSHU                61     2016/05/09    2016/05/09 08:02:37  FRED
........ X04SXPU                66     2016/05/09    2016/05/09 08:02:38  FRED
........ X05CSFS                50     2016/05/09    2016/05/09 08:02:39  FRED
........ X06SRVA                28     2016/05/09    2016/05/09 08:02:39  FRED
........ X07SAVE                33     2016/05/09    2016/05/09 08:02:40  FRED
........ X08ZFSA                70     2016/05/09    2016/05/09 08:02:40  FRED
........ X09LOAD                96     2016/05/09    2016/05/09 08:02:41  FRED
........ X10REST                26     2016/05/09    2016/05/09 08:02:41  FRED
........ X11OPTT                22     2016/05/09    2016/05/09 08:02:42  FRED
........ X12LIBS                79     2016/05/09    2016/05/09 08:02:47  FRED
........ X13PROC                32     2016/05/09    2016/05/09 08:02:49  FRED
........ X14LICN                39     2016/05/09    2016/05/09 08:02:50  FRED
........ X15KEYG                23     2016/05/09    2016/05/09 08:02:50  FRED
         **End**
```

Figure 2.6. Tectia SSH Assistant Member list of generated installation jobs (2.1 JOBS). The job names in this screen capture are based on using the default job card.

5.   Log off and on.

6.   EXEC the Tectia SSH Assistant application.

     To start the SSH server, go to **4 TASK → 1 TSRV** and enter **1 TSRVS**.

     To check the version of the running server, enter **5 TSRVQV**.

2.3.5 Enabling Manual Pages

The online manual pages for the product are installed to /opt/tectia/man. If you want your USS manual page tool (/bin/man) to be able to find the new manual pages, you will have to perform the following environment setup.

If you want to have the manual pages visible to all USS shell users, you must edit system-wide configuration files:

1.   Add the following lines to file /etc/profile:

```
if test X$MANPATH = X ; then
  MANPATH="/usr/man/%L"
fi
MANPATH="/opt/tectia/man/%L:$MANPATH"
export MANPATH
```

2.   Add the following lines to file /etc/cshrc:

```
if
(! $?MANPATH) setenv MANPATH "/usr/man/%L"
setenv MANPATH "/opt/tectia/man/%L:$MANPATH"
```

If you want to have the manual pages visible to just one user, you must edit two files in the user's home directory:

1. Add the following lines to `.profile` in the user's home directory:

```
if test X$MANPATH = X ; then
  MANPATH="/usr/man/%L"
fi
MANPATH="/opt/tectia/man/%L:$MANPATH"
export MANPATH
```

2. Add the following lines to file `.cshrc` in the user's home directory:

```
if (! $?MANPATH) setenv MANPATH "/usr/man/%L"
setenv MANPATH "/opt/tectia/man/%L:$MANPATH"
```

2.4 Removing the Tectia Server for IBM z/OS Software

To remove Tectia Server for IBM z/OS, run jobs `U01ZDEL`, `U02UNIN` and `U03DELU` in the Tectia SSH Assistant submenu **5 UTIL** ("Utility jobs to manage the installation").

Figure 2.7. Tectia SSH Assistant Utility jobs to manage the installation (5 UTIL)

The three jobs achieve the following:

1. `U01ZDEL` - Delete the previous installation ZFS

    ⚠ **Caution**

    Use this job only if you are sure the ZFS is no longer needed, that is, if the product is to be uninstalled or has been successfully upgraded into a different ZFS.

2. `U02UNIN` - Delete the SSZ installation dataset

⚠️ **Caution**

> This job will delete all data set components of the SSZ installation: started task procedures, samples, parms, as well as the `/opt/tectia` symlink.

3. `U03DELU` - Delete the SSH server and SOCKS Proxy proxy users

Chapter 3 Getting Started with Tectia Server for IBM z/OS

This chapter provides information on how to get started with Tectia Server for IBM z/OS after it has been successfully installed.

It contains information on running the server, the required environment variables, and setting up a shell user.

> **Note**
>
> To access a z/OS host that runs Tectia Server using OpenSSH **scp** you must either have OpenSSH server running on the same z/OS host with Tectia Server, or OpenSSH **scp** must exist in `/bin/scp` on the z/OS host.

> **Note**
>
> If you intend to use OpenSSH **scp** with Tectia Server for IBM z/OS, note that the default OpenSSH configuration on z/OS does not produce SMF records. SMF recording must be configured separately for OpenSSH when the OpenSSH **scp** events need to be captured.

The Tectia server component on z/OS consists of two processes:

- **sshd2**: the main Secure Shell server daemon

- **ssh-certd**: the Certificate Validator, a process used by **sshd2** when validating user certificates

3.1 Running the SSH Server (sshd2)

3.1.1 Starting the Server

Console

To run **sshd2** as a started task, use a JCL procedure such as SSHD2 (shown below), by default USER.PRO-
CLIB(SSHD2) (defined in 0.2 SETO).

SSHD2:

```
//SSHD2   PROC OPTS='',PORT=
//TECTIA  EXEC PGM=BPXBATSL,
//             REGION=0M,
//             TIME=NOLIMIT,
//             PARM=('PGM /opt/tectia/sbin/sshd2 -F &PORT    ❶
//              &OPTS')
//STDENV   DD  DSN=<HLQ>.V6611.PARMLIB(SSHENV),DISP=SHR
//STDOUT   DD  SYSOUT=*
//*STDERR   DD  SYSOUT=*
//STDIN    DD  DUMMY
//         PEND
```

❶   Note that when **sshd2** is run as a started task, the -F (foreground) option is required to prevent **sshd2**
      from detaching itself as a daemon. If **sshd2** is run as a started task without the -F option, it cannot be
      modified.

Start the server with the following operator command:

```
===> s sshd2
```

The sshd2 job starts.

ISPF

In the Tectia SSH Assistant ISPF application you can manage SSH tasks via submenu **4 TASK**. Started tasks
may be started, stopped and modified here, provided that the user is properly authorized and that the started
task procedure has been generated and installed.

You can control the SSH server (**sshd2**) via submenu **4.1 TSRV**.

Figure 3.1. Tectia SSH Assistant ISPF application - Tasks: SSH Server (4.1 TSRV)

To start the server, enter option **4.1.1 TSRVS (Start the SSH server)**.

You should see the following console message:

```
 ISF031I CONSOLE <USERID> ACTIVATED
-S SSHD2
 +SSZ0006I Task sshd2 started
```

USS

> **ℹ Note**
>
> If you want to run the server manually from USS, the SSHD2 user needs to have a login shell defined.
> Modify the ADDSSHD2 script accordingly.

To start the server manually, log on to Unix System Services (USS) as the SSHD2 user and execute the command:

```
> /opt/tectia/etc/init.d/sshd2 start
```

When the version string message appears, you may exit from the shell. The server will continue to run as a process without a controlling terminal. During the startup the server might report warnings that the server daemon cannot access the cryptographic hardware. Those are only warnings and the server starts without the cryptographic hardware mentioned in the warning messages.

3.1.2 Stopping the Server

Console

To stop the server under MVS when you are running it as a started task, enter:

```
===> F SSHD2,STOP
```

OR:

```
===> P SSHD2
```

ISPF

To stop the server in ISPF, enter the Tectia SSH Assistant option **4.1.2 TSRVP (Stop the SSH server)**.

You should see the following console message:

```
 ISF031I CONSOLE <USERID> ACTIVATED
-P SSHD2
 +SSZ0003I Command STOP accepted
 +SSZ0005I Dispatching signal to S=<PID> (sshd2)
```

USS

Under USS, if you want just to stop the server (without restarting it), you can use the following command:

```
> /opt/tectia/etc/init.d/sshd2 stop
```

3.1.3 Restarting the Server

Console

To restart the server under MVS when you are running it as a started task, use the following console command:

```
===> F SSHD2,RESTART
```

When the server is restarted, existing connections will stay open until they are disconnected. If you have made configuration changes, existing connections will continue to use the old configuration settings while new connections will use the reconfigured settings. To ensure that the new settings will be used for all new connections, use the Tectia **sshg3** client option `--exclusive`. This way a new connection is opened for each connection attempt, instead of the Connection Broker reusing recently closed connections.

To kill existing connections and restart the server, use the **restart** command's `force` option:

```
===> F SSHD2,RESTART FORCE
```

User address spaces that were running before the stop will continue to run. Any changes to the configuration or keys will take effect when the daemon is restarted and when new user connections are established.

ISPF

To restart the server in ISPF, enter the Tectia SSH Assistant option **4.1.3 TSRVR (Restart the SSH server)**.

You should see the following console message:

```
 ISF031I CONSOLE <USERID> ACTIVATED
-F SSHD2,RESTART
 +SSZ0003I Command RESTART accepted
```

To kill existing connections and restart the server, enter the Tectia SSH Assistant option **4.1.4 TSRVRF (Restart the SSH server, killing connections)**.

3.1.4 Querying the Server Version

Console

You can query the version of the server with the following console command:

```
===> F SSHD2,VERSION
```

ISPF

To check the version of the server in ISPF, enter the Tectia SSH Assistant option **4.1.5 TSRVQV (Query the version of the running SSH server)**.

You should see the following console messages:

```
 ISF031I CONSOLE <USERID> ACTIVATED
-F SSHD2,VERSION
 +SSZ0003I Command VERSION accepted
 +SSZ0004I Tectia Server for z/OS <version>
  SSZ0008I VC branch: <branch>
  SSZ0009I VC revision: <revision>
```

> **Note**
>
> If some of the console messages fail to show up, you can check them in the SDSF log.

3.1.5 Setting Options for Starting the Server

Console

You can enter command options (described in detail in sshd2 Options) as an OPTS parameter on the **start**. For example, configuration file options are entered in the following format (where `keyword` is a configuration file keyword):

```
===> S SSHD2,OPTS='-Okeyword=value'
```

For example, to set `SftpSmfType` to `TYPE119`, enter:

```
===> S SSHD2,OPTS='-oSftpSmfType=TYPE119'
```

The `sshd2` started task can also be started with a user-specified job name:

```
===> s SSHD2,jobname=own_job_name
```

You can assign the `SSHD2` user to the started task by defining the procedure in the `STARTED` class and entering the user ID in the `STDATA` segment, for example:

```
RDEFINE STARTED SSHD2.* STDATA(USER(SSHD2)GROUP(SYS1))
SETROPTS RACLIST(STARTED) REFRESH
```

ISPF

To set parameter options for starting the SSH server in ISPF, use the Tectia SSH Assistant option **4.1.7 TSR-VOP (Set options for starting the SSH server)**.

The available options are described in detail in sshd2 Options.

The options you enter will be used when the server is started.
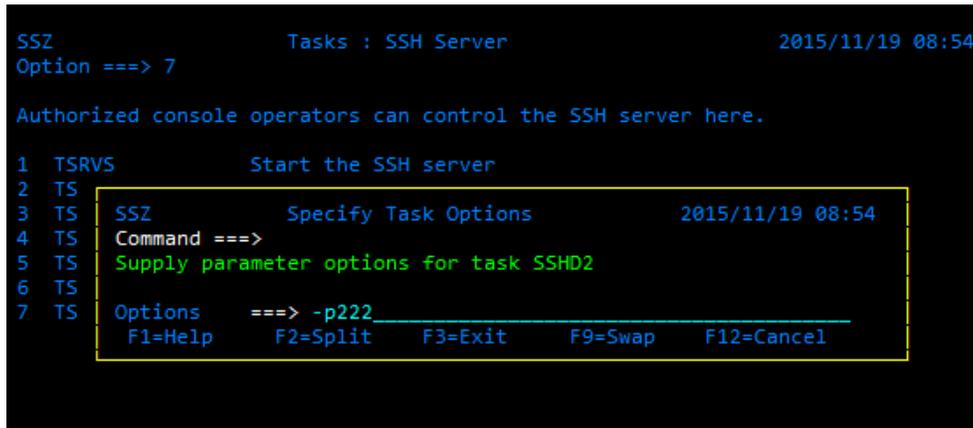
Figure 3.2. Tectia SSH Assistant ISPF application - Setting port number

3.2 Running the Certificate Validator (ssh-certd)

3.2.1 Starting the Certificate Validator

Console

To run **ssh-certd** as a started task, use a JCL procedure such as SSHCERTD (shown below), by default USER.PROCLIB(SSHCERTD) (defined in 0.2 SETO).

SSHCERTD:

```
//SSHCERTD PROC OPTS=''
//TECTIACD EXEC PGM=BPXBATSL,
//            REGION=0M,
//            TIME=NOLIMIT,
//            PARM=('PGM /opt/tectia/sbin/ssh-certd -F
//             &OPTS')
//STDENV   DD  DSN=<HLQ>.V6611.PARMLIB(SSHENV),DISP=SHR
//STDOUT   DD  SYSOUT=*
```

```
//*STDERR   DD  SYSOUT=*
//STDIN     DD  DUMMY
//          PEND
```

Start the Certificate Validator with the following operator command:

```
===> S SSHCERTD
```

In the sample SSHCERTD script above, **ssh-certd** is started with the foreground option that disables the daemon mode. With the foreground option, the daemon does not spawn the process to background and the task name stays as sshcertd.

You can assign the SSHD2 user to the started task by defining the procedure in the STARTED class and entering the user ID in the STDATA segment, for example:

```
RDEFINE STARTED SSHCERTD.* STDATA(USER(SSHD2)GROUP(SYS1))
SETROPTS RACLIST REFRESH
```

For more information, see ssh-certd(8).

ISPF

You can control the Certificate Validator (**ssh-certd**) via the Tectia SSH Assistant submenu **4.2 TCRT**.
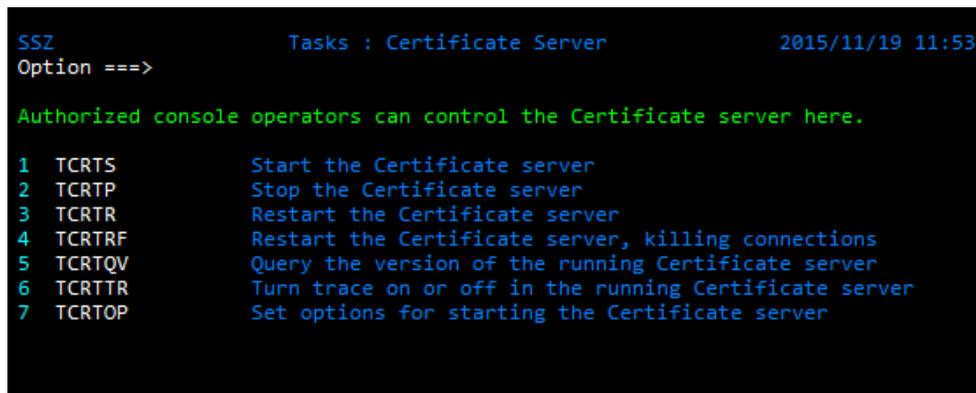
Figure 3.3. Tectia SSH Assistant ISPF application - Tasks: Certificate Server (4.2 TCRT)

To start the Certificate Validator, enter option **4.2.1 TCRTS (Start the Certificate server)**.

You should see the following console message:

```
 ISF031I CONSOLE <USERID> ACTIVATED
-S SSHCERTD
 +SSZ0006I Task ssh-certd started
```

3.2.2 Stopping the Certificate Validator

Console

To stop the Certificate Validator (without restarting it) under MVS when you are running it as a started task, enter the following console command:

```
===> F SSHCERTD,STOP
```

OR:

```
===> P SSHCERTD
```

ISPF

To stop the Certificate Validator in ISPF, enter the Tectia SSH Assistant option **4.2.2 TCRTP (Stop the Certificate server)**.

You should see the following console message:

```
 ISF031I CONSOLE <USERID> ACTIVATED
-P SSHCERTD
 +SSZ0003I Command STOP accepted
 +SSZ0005I Dispatching signal to S=<PID> (ssh-certd)
```

3.2.3 Restarting the Certificate Validator

Console

To restart the Certificate Validator under MVS when you are running it as a started task, enter the following:

```
===> F SSHCERTD,RESTART
```

ISPF

To restart the Certificate Validator in ISPF, enter the Tectia SSH Assistant option **4.2.3 TCRTR (Restart the Certificate server)**.

You should see the following console message:

```
 ISF031I CONSOLE <USERID> ACTIVATED
-F SSHCERTD,RESTART
 +SSZ0003I Command RESTART accepted
```

3.2.4 Querying the Certificate Validator Version

Console

To query the version of the running Certificate Validator, enter the following console command:

```
===> F SSHCERTD,VERSION
```

ISPF

To query the version of the Certificate Validator in ISPF, enter the Tectia SSH Assistant option **4.2.5 TCRTQV (Query the version of the running Certificate server)**.

3.2.5 Setting Options for Starting the Certificate Validator

Console

As an OPTS parameter, you can give parameters that the actual Certificate Validator binary accepts (described in detail in ssh-certd Options). For example:

```
===> S SSHCERTD,OPTS='-d 9'
```

ISPF

To set options for starting the Certificate Validator in ISPF, enter the Tectia SSH Assistant option **4.2.7 TCRTOP**.

The available options are described in detail in ssh-certd Options.

3.3 Environment Variables for Server and Client Applications

The environment variables `_BPXK_AUTOCVT`, `_BPX_SHAREAS` and `_BPX_BATCH_UMASK` must be set as shown in `<HLQ>.V6611.PARMLIB(SSHENV)` and `sshsetenv` (located in `/opt/tectia/doc/zOS/samples`) when running Tectia Server for IBM z/OS programs (see below). The server startup procedures described in Section 3.1 set these variables from the STDENV DD.

The format of `SSHENV` is used when the client or server programs are run under MVS, and `sshsetenv` is used when the programs are run from a USS command line.

> ### ℹ️ Note
>
> The environment files must not contain line numbers or reading them will fail.

SSHENV:

```
_BPXK_AUTOCVT=ON
_BPX_SHAREAS=NO
_BPX_BATCH_UMASK=0022
SSH_DEBUG_FMT="%W(72)(2) %Dd/%Dt/%Dy %Dh:%Dm:%Ds:%Df %m/%s:%n:%f %M"
_BPXK_JOBLOG=STDERR
_EDC_ADD_ERRNO2=1
```

sshsetenv:

```
export _BPXK_AUTOCVT=ON
export _BPX_BATCH_UMASK=0022
export _BPX_SHAREAS=NO
export SSH_DEBUG_FMT="%W(72)(2) %Dd/%Dt/%Dy %Dh:%Dm:%Ds:%Df %m/%s:%n:%f %M"
export _EDC_ADD_ERRNO2=1
```

The server will do coded character set conversion on request when transferring files. The `iconv` function is used for the conversion. `iconv()` uses the following environment variables (it is usually not necessary to change them):

_ICONV_UCS2

Tells `iconv_open(Y, X)` which type of a conversion method to set up when there is a choice between "direct" conversion from X to Y and "indirect" X to UCS-2 to Y.

_ICONV_UCS2_PREFIX

Tells `iconv_open()` which z/OS data set name prefix to use to find the UCS-2 tables if they cannot be found in the HFS.

3.4 Setting Up a Shell User

User accounts that are accessed via Tectia Server or that run Tectia client programs must have an OMVS segment in their RACF profile and, in most cases, a home directory in the USS file hierarchy. The home directory is required if public key authentication is used and if user-specific configuration is needed.

z/OS users that are going to use the Tectia Server for IBM z/OS client programs to access remote hosts need the same OMVS environment. For an example of creating a z/OS user for batch file transfers, see Section 7.1.

To enable the user to open the shell or other daemon, you need to increase the maximum region size (in bytes) for address space of the WSA process. You can set a system-wide limit in BPXPRMxx and then set higher limits for individual processes.

For OMVS, increase the size of the address space to 75MB in the RACF user profile. Use the RACF ADDUSER or ALTUSER command to specify the ASSIZEMAX limit on a per-process basis as follows:

```
ALTUSER userid OMVS(ASSIZEMAX(75000000)
```

In batch jobs, `REGION=` sets the address space size. It can be used on the JOB statement or on the EXEC statement.

If you want to customize the environment per user for file transfer use, see *Tectia Server for IBM z/OS User Manual*.

3.4.1 Authenticating Remote Server Hosts

Remote Secure Shell servers are authenticated by a public-key procedure. The client trusts the server if it has a private key that matches one of the public keys in the global hostkeys directory or in the user's hostkeys directory. When a server presents a key that is not in the hostkeys directories, the user checks the fingerprint of the remote server's public key. When the user has approved the public key, it is stored in the user's `$HOME/.ssh2/hostkeys` directory and will be used automatically thereafter.

The verification step requires user interaction, so even for users that are set up to run client programs unattended, the first connection must be done by a person who logs in as the user, accesses the remote server, and goes through the fingerprint check dialog. The same steps must be repeated if the remote host's key is changed.

For more information on server authentication, see Section 5.3 and *Tectia Server for IBM z/OS User Manual*.

3.4.2 Using Password Authentication

Password authentication is the most commonly used form of user authentication. It is enabled by default and uses the RACF system password of the user.

Note that when running Tectia client programs from the TSO OMVS pseudo-shell, the password that you type in will be shown on your screen in plaintext. You can avoid showing the password on your screen by invoking OMVS as follows:

```
TSO OMVS PF13(HIDE)
```

Start your client (in this example **sftpg3**) to request password:

```
sftpg3 --aa password hostname
username@hostname's password:
```

Press **P13** before typing in your password! This will temporarily hide the input data you type on the shell command line.

Other ways to avoid showing your password on the screen are using a real shell via telnet or SSH, or setting up public-key authentication using a private key without a passphrase.

There is one case where password authentication cannot be used in Tectia Server for IBM z/OS: When running Tectia client programs from JCL there is no facility for getting the password interactively from the user. You can have the password stored in a file or a data set, or preferably use public-key authentication and a private key without a passphrase (see Section 5.6).

For more information on password authentication, see Section 5.5 and *Tectia Server for IBM z/OS User Manual*.

3.4.3 Using Public-Key Authentication

In public-key authentication, the server authenticates the user by the presence of the user's public key in the user's $HOME/.ssh2 directory on the server. The public key ties the user ID to the user's private key stored on the client. Keys can be generated using the **ssh-keygen-g3** tool.

For more information on public-key authentication, see Section 5.6 and *Tectia Server for IBM z/OS User Manual*.

Chapter 4 Configuring the Server

This chapter gives instructions on the basic configuration settings of the Tectia server component on z/OS.

> **ⓘ Note**
>
> The server must be restarted after configuration changes. See Section 3.1.3 for instructions.

4.1 Server Configuration Files

The following files and directories located in the `/opt/tectia/etc` directory are used to store the server configuration information:

- `sshd2_config`: the main server configuration file

- `ssh_certd_config`: the Certificate Validator configuration file

- `hostkey`: the default server host private key

- `hostkey.pub`: the default server host publickey

- `random_seed`: the random number seed file for cryptographic operations

- `knownhosts`: the global directory for remote client host public keys that are trusted for host-based authentication

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

- `knownhosts`: the user-specific directory for remote client host public keys that are trusted for host-based authentication

- `authorization`: the default authorization file for user public keys

4.1.1 Editing Configuration Files

The default location for the configuration files is `/opt/tectia/etc`. You can edit the `sshd2_config` and `ssh_certd_config` files via the Tectia SSH Assistant or with your favorite text editor.
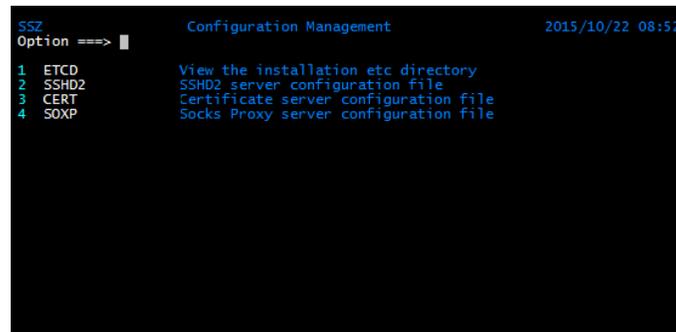

```
SSZ                    Configuration Management              2015/10/22 08:52
Option ===> █

1   ETCD               View the installation etc directory
2   SSHD2              SSHD2 server configuration file
3   CERT               Certificate server configuration file
4   SOXP               Socks Proxy server configuration file
```

Figure 4.1. Tectia SSH Assistant Configuration Management menu (3 CONF)

The configuration files consist of keyword-value pairs, one per line. Lines starting with the hash character (#) as well as empty lines are interpreted as comments and ignored. The configuration file is read when the server is started and each time a new connection to the server is made. If the configuration file is faulty or cannot be found, the server will not start.

For a complete list of configuration options, see sshd2_config(5) and ssh_certd_config(5).

4.1.2 Command-Line Options

Command-line options can be used to configure the server in addition to the configuration file and environment variables. Command-line options override values specified in environment variables and the configuration file.

For a list of the available options, see sshd2(8).

4.1.3 Running Tectia and OpenSSH on the Same Host

Tectia Server for IBM z/OS always installs under `/opt/tectia`, so the Tectia and OpenSSH binaries can coexist in the system. The Tectia configuration files and host keys are stored in the `/opt/tectia/etc` directory, whereas OpenSSH uses the `/etc/ssh` directory. The Tectia user keys and other user-specific files are stored under `$HOME/.ssh2`, whereas OpenSSH uses `$HOME/.ssh`. The server port can be configured in the `/opt/tectia/etc/sshd2_config` file with the keyword **Port**. The default port number is 22, but it can be changed to any available port.

4.1.4 IPv6 Support

Tectia Server for IBM z/OS can operate using IPv4 (`inet`) addressing, IPv6 (`inet6`), or both (`any`), as specified by the `sshd2_config` keyword **AddressFamily**:

```
AddressFamily                    (inet|inet6|any)
```

The default is `inet`.

The **ListenAddress** keyword can be specified several times. The value is an IPv4 address or an IPv6 address; either may be followed by an optional port number. If the configuration variable is not specified, **sshd2** listens on the IPv4 unspecified address (`0.0.0.0`) or the IPv6 unspecified address (`::`) or both depending on the value of `AddressFamily`.

sshd2 will open listener sockets in the order the `ListenAddress` configuration variables occur in the configuration file. `sshd2` will not start unless it is able to open at least one socket. `sshd2` will not start if any of the sockets cannot be opened or does not belong to the address family specified in `AddressFamily` or by the `-4` or `-6` command line option.

4.2 Defining Subconfigurations

Subconfiguration files can be used to specify configuration options that apply only to connections by specific users or from specific hosts. The subconfiguration files have the same basic format as the main configuration file and they are divided into two categories: host-specific and user-specific.

If parsing of the subconfiguration files fails, the connection is terminated (for a host-specific configuration), or the access is denied by the server (for a user-specific configuration).

Most of the configuration options that work in the main file work also in the subconfiguration files, but some do not, where it either does not make sense to set them (for example, `ListenAddress` and `Port`, which only affect the process listening to the port, and would not affect that behavior in any way in a subconfiguration file) or it would be confusing (e.g. `AllowUsers` in a user-specific subconfiguration, and `AllowHosts` in a host-specific subconfiguration).

The value for {`Host`,`User`}`SpecificConfig` keywords is a pattern-filename pair, separated by whitespace. With `UserSpecificConfig`, the pattern is of format `user[%group][@host]`, where `user` is matched with the user name and UID, `group` is matched with the user's primary and any secondary groups, both group name and GID, and `host` is matched as described under option `AllowHosts`. With `HostSpecificConfig`, the pattern is `host` (as in `UserSpecificConfig`).

Unlike the main configuration file, the subconfiguration files may have configuration blocks, or stanzas, in them. The subconfiguration heading is interpreted identically to what is described above, that is with `User-SpecificConfig` the pattern is of format `user[%group][@host]`, and with `HostSpecificConfig` the format is `host`.

> **ℹ️ Note**
>
> It is possible to mix these configuration files. This is not recommended, because any global settings in these files would be set multiple times (which would not do any harm per se, but might lead to behavior not intended by the administrator).

Subconfiguration files are very flexible and because of that, dangerous if the logic of the files is not carefully planned. You can, for example, specify different authentication methods for different users and different banner messages for people coming from certain hosts. There are a lot of possibilities here.

> **ℹ️ Note**
>
> Host-specific subconfiguration files are always read before the user-specific subconfiguration files. See the example file `/opt/tectia/etc/sshd2_config.example` and the host-specific and user-specific files in `/opt/tectia/etc/subconfig`.

4.2.1 Host-Specific Subconfiguration

The host-specific configuration files are configured with the `HostSpecificConfig` variable. These files are read immediately after a new process is launched to handle the connection. Thus most configuration options can be set in these. The syntax is the following:

```
HostSpecificConfig pattern subconfig-file
```

`pattern` will be used to match the client host as specified under **AllowHosts** on the sshd2_config(5) man page. The file `subconfig-file` will then be read, and configuration data amended accordingly.

The file is read before any actual protocol transactions begin, and you can specify most of the options allowed in the main configuration file. You can specify more than one subconfiguration file, in which case the patterns are matched and the files read in the specified order. Values of configuration options defined later will either override or amend the previous value depending on the option. The effect of redefining an option is described in the documentation for that option. For example, setting **Ciphers** in the subconfiguration file will override the old value, but setting **AllowUsers** will amend the value.

Example 1: The following matches (from) any host:

```
HostSpecificConfig   .* /opt/tectia/etc/subconfig/host_ext.conf
```

Example 2: The following matches a subnet mask:

```
HostSpecificConfig   \m192.168.0.0/16   /opt/tectia/etc/subconfig/host_int.conf
```

For more information, please see sshd2_subconfig(5) and sshd2_config(5).

4.2.2 User-Specific Subconfiguration

User-specific subconfiguration files are read when the client has stated the user name it is trying to log in as. At this point, the server will obtain additional information about the user: does the user exist, what is the user's UID, and what groups the user belongs to. With this information, the server can read the user-specific configuration files specified with **UserSpecificConfig** in the main configuration file. The syntax is the following:

```
UserSpecificConfig pattern subconfig-file
```

You can use patterns of the following form:

```
user[%group][@host]
```

where `user` is matched with the user name and UID, `group` is matched with the user's primary and secondary groups, both group name and GID, and `host` is matched as described under **AllowHosts** on the sshd2_config(5) man page.

For example, the following would match any user in group "sftp" connecting from example.com:

```
.*%sftp@example\.com
```

Example 1: The following matches to users from `ssh.com` who have two-character user names or the user name `sjl`, and who belong to the group `wheel`.

```
UserSpecificConfig  (..|sjl)%wheel@ssh\.com    /opt/tectia/etc/subconfig/user_conf
```

Example 2: The following matches the user `anon` from any host:

```
UserSpecificConfig  anon@.*   /opt/tectia/etc/subconfig/anon_conf
```

See the sshd2_subconfig(5) man page for more information.

4.3 Configuring Cryptographic Algorithms

Note

Algorithm names are case-sensitive.

4.3.1 Configuring Ciphers

The algorithm(s) used for session encryption can be specified in the sshd2_config file:

```
Ciphers              aes128-cbc,3des-cbc
```

Currently supported cipher names are the following:

```
aes128-ctr              aes128-cbc              twofish-cbc
aes192-ctr              aes192-cbc              twofish128-cbc
```

aes256-ctr	aes256-cbc	twofish192-cbc
aes128-gcm@openssh.com	3des-cbc	twofish256-cbc
aes256-gcm@openssh.com	arcfour	cast128-12-cbc@ssh.com
aes128-gcm@openssh.com	blowfish-cbc	seed-cbc@ssh.com
aes256-gcm@openssh.com	cast128-cbc	rijndael-cbc@ssh.com

Special values for this option are the following:

- `Any`: includes all supported ciphers plus `none`.

- `AnyStd`: includes ciphers from the IETF SSH standards and `none`. The standard ciphers are `aes128-ctr`, `aes192-ctr`, `aes256-ctr`, `aes128-gcm@openssh.com`, `aes256-gcm@openssh.com`, `aes128-cbc`, `aes192-cbc`, `aes256-cbc`, `3des-cbc`, `arcfour`, `blowfish-cbc`, `cast128-cbc`, `twofish128-cbc`, `twofish192-cbc`, `twofish256-cbc`, `twofish-cbc`.

- `none`: no encryption, connection will be in plaintext.

- `AnyCipher`: allows any available cipher apart from the non-encrypting cipher mode `none`.

- `AnyStdCipher`: the same as `AnyStd`, but excludes `none`.

The default ciphers are `aes128-ctr`, `aes192-ctr`, `aes256-ctr`, `aes128-gcm@openssh.com`, `aes256-gcm@openssh.com`, `aes128-cbc`, `aes192-cbc`, `aes256-cbc` and `3des-cbc`.

4.3.2 Configuring MACs

The MAC (Message Authentication Code) algorithm(s) used for data integrity verification can be selected in the `sshd2_config` file:

```
MACs                hmac-sha1,hmac-md5
```

The supported MAC names are the following:

hmac-md5	hmac-sha2-256	hmac-sha384@ssh.com
hmac-md5-96	hmac-sha256-2@ssh.com	hmac-sha2-512
hmac-sha1	hmac-sha224@ssh.com	hmac-sha512@ssh.com
hmac-sha1-96	hmac-sha256@ssh.com	

Special values for this option are the following:

- `Any`: includes all supported MACs plus `none`.

- `AnyStd`: includes MACs from the IETF SSH standards (`hmac-md5`, `hmac-md5-96`, `hmac-sha1`, `hmac-sha1-96`, `hmac-sha2-256`, `hmac-sha2-512`) and `none`.

- `none`: means that no cryptographic data integrity method is used.

- `AnyMac`: the same as `Any` but excludes `none`.

- `AnyStdMac`: the same as `AnyStd` but exludes `none`.

The default MAC algorithms are: `hmac-sha1`, `hmac-sha1-96`, `hmac-sha2-256`, `hmac-sha256-2@ssh.com`, `hmac-sha224@ssh.com`, `hmac-sha256@ssh.com`, `hmac-sha384@ssh.com`, `hmac-sha2-512`, and `hmac-sha512@ssh.com`.

4.3.3 Configuring KEXs

The key exchange (KEX) algorithm(s) used for key exchange can be selected in the `sshd2_config` file. Multiple KEXs can be specified as a comma-separated list.

```
KEXs     diffie-hellman-group14-sha1,diffie-hellman-group14-sha224@ssh.com
```

The supported KEX algorithms are the following:

`diffie-hellman-group1-sha1`	`diffie-hellman-group15-sha384@ssh.com`
`diffie-hellman-group14-sha1`	`diffie-hellman-group16-sha384@ssh.com`
`diffie-hellman-group14-sha256`	`diffie-hellman-group16-sha512@ssh.com`
`diffie-hellman-group16-sha512`	`diffie-hellman-group18-sha512@ssh.com`
`diffie-hellman-group18-sha512`	`ecdh-sha2-nistp256`
`diffie-hellman-group14-sha224@ssh.com`	`ecdh-sha2-nistp384`
`diffie-hellman-group14-sha256@ssh.com`	`ecdh-sha2-nistp521`
`diffie-hellman-group15-sha256@ssh.com`	

Special values for this option are the following:

- `Any`: includes all supported KEX algorithms.

- `AnyStd`: includes the following KEXs from the IETF SSH standards: `ecdh-sha2-nistp256`, `ecdh-sha2-nistp384`, `ecdh-sha2-nistp521`, `diffie-hellman-group14-sha1`, and `diffie-hellman-group1-sha1`.

- `AnyKEX`: the same as `Any`.

- `AnyStdKEX`: the same as `AnyStd`.

The default KEX algorithms are: `ecdh-sha2-nistp521`, `ecdh-sha2-nistp384`, `ecdh-sha2-nistp256`, `diffie-hellman-group14-sha1`, `diffie-hellman-group14-sha256`, `diffie-hellman-group16-sha512`, `diffie-hellman-group18-sha512`, `diffie-hellman-group14-sha256@ssh.com`.

4.3.4 Configuring Host Key Signature Algorithms

The host key signature algorithms to be used in server authentication and host-based authentication can be selected in the `sshd2_config` file using the `HostKeyAlgorithms` keyword. The keyword defines the host key signature algorithms that the server will propose and accept to authenticate the host. Using the keyword, it is possible to enable only certain hash functions, such as SHA-2. A message is signed with a hash generated

using a signature algorithm and then verified by the receiver using the same signature algorithm. Multiple host key algorithms can be specified as a comma-separated list.

```
HostKeyAlgorithms                ssh-dss-sha224@ssh.com
```

The client defines the order of host key signature algorithms. The client should have at least one algorithm in common with the server configuration. The supported signature algorithms are the following:

rsa-sha2-256	ssh-rsa-sha256@ssh.com
rsa-sha2-512	ssh-rsa-sha384@ssh.com
ssh-dss	ssh-rsa-sha512@ssh.com
ssh-dss-sha224@ssh.com	x509v3-sign-rsa
ssh-dss-sha256@ssh.com	x509v3-sign-rsa-sha224@ssh.com
ssh-dss-sha384@ssh.com	x509v3-sign-rsa-sha256@ssh.com
ssh-dss-sha512@ssh.com	x509v3-sign-rsa-sha384@ssh.com
x509v3-sign-dss	x509v3-sign-rsa-sha512@ssh.com
x509v3-sign-dss-sha224@ssh.com	ecdsa-sha2-nistp256
x509v3-sign-dss-sha256@ssh.com	ecdsa-sha2-nistp384
x509v3-sign-dss-sha384@ssh.com	ecdsa-sha2-nistp521
x509v3-sign-dss-sha512@ssh.com	x509v3-ecdsa-sha2-nistp256
ssh-rsa	x509v3-ecdsa-sha2-nistp384
ssh-rsa-sha224@ssh.com	x509v3-ecdsa-sha2-nistp521

Special values for this option are the following:

- `Any`: includes all supported host key signature algorithms.

- `AnyStd`: includes the following signature algorithms from the IETF SSH standards: `x509v3-sign-dss`, `x509v3-sign-rsa`, `ssh-dss`, `ssh-rsa`, `x509v3-ecdsa-sha2-nistp521`, `x509v3-ecdsa-sha2-nistp384`, `x509v3-ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp521`, `ecdsa-sha2-nistp384`, and `ecdsa-sha2-nistp256`.

- `AnyHostKeyAlgorithm`: the same as `Any`.

- `AnyStdHostKeyAlgorithm`: the same as `AnyStd`.

The default host key signature algorithms are:

ecdsa-sha2-nistp256	ssh-rsa-sha256@ssh.com
ecdsa-sha2-nistp384	ssh-dss
ecdsa-sha2-nistp521	ssh-dss-sha256@ssh.com
x509v3-ecdsa-sha2-nistp256	x509v3-sign-dss
x509v3-ecdsa-sha2-nistp384	x509v3-sign-dss-sha256@ssh.com
x509v3-ecdsa-sha2-nistp521	x509v3-sign-rsa
ssh-rsa	x509v3-sign-rsa-sha256@ssh.com

4.3.5 Configuring Public Key Signature Algorithms

The public key signature algorithms to be used in client authentication can be selected in the `sshd2_config` file using the `AuthPublicKey.Algorithms` keyword. The keyword defines the public key signature algorithms that the server will propose and accept to authenticate the user. Using the keyword, it is possible to enable only certain hash functions, such as SHA-2. A message is signed with a hash generated using a signature algorithm and then verified by the receiver using the same signature algorithm. Multiple public key algorithms can be specified as a comma-separated list.

```
AuthPublicKey.Algorithms          ssh-dss-sha224@ssh.com
```

The client defines the order of public key signature algorithms. The client should have at least one algorithm in common with the server configuration. The supported signature algorithms are the following:

```
rsa-sha2-256                              ssh-rsa-sha256@ssh.com
rsa-sha-512                               ssh-rsa-sha384@ssh.com
ssh-dss                                   ssh-rsa-sha512@ssh.com
ssh-dss-sha224@ssh.com                    x509v3-sign-rsa
ssh-dss-sha256@ssh.com                    x509v3-sign-rsa-sha224@ssh.com
ssh-dss-sha384@ssh.com                    x509v3-sign-rsa-sha256@ssh.com
ssh-dss-sha512@ssh.com                    x509v3-sign-rsa-sha384@ssh.com
x509v3-sign-dss                           x509v3-sign-rsa-sha512@ssh.com
x509v3-sign-dss-sha224@ssh.com            ecdsa-sha2-nistp256
x509v3-sign-dss-sha256@ssh.com            ecdsa-sha2-nistp384
x509v3-sign-dss-sha384@ssh.com            ecdsa-sha2-nistp521
x509v3-sign-dss-sha512@ssh.com            x509v3-ecdsa-sha2-nistp256
ssh-rsa                                   x509v3-ecdsa-sha2-nistp384
ssh-rsa-sha224@ssh.com                    x509v3-ecdsa-sha2-nistp521
```

Special values for this option are the following:

- `Any`: includes all supported signature algorithms.

- `AnyStd`: includes the following signature algorithms from the IETF SSH standards: `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, `ecdsa-sha2-nistp521`, `x509v3-ecdsa-sha2-nistp256`, `x509v3-ecdsa-sha2-nistp384`, `x509v3-ecdsa-sha2-nistp521`, `x509v3-sign-dss`, `x509v3-sign-rsa`, `ssh-dss`, and `ssh-rsa`.

- `AnyPublicKeyAlgorithm`: the same as `Any`.

- `AnyStdPublicKeyAlgorithm`: the same as `AnyStd`.

The default public key signature algorithms are:

```
rsa-sha2-256                              ssh-rsa
rsa-sha2-512                              ssh-rsa-sha256@ssh.com
```

```
ecdsa-sha2-nistp256                          ssh-dss
ecdsa-sha2-nistp384                          ssh-dss-sha256@ssh.com
ecdsa-sha2-nistp521                          x509v3-sign-dss
x509v3-ecdsa-sha2-nistp256                   x509v3-sign-dss-sha256@ssh.com
x509v3-ecdsa-sha2-nistp384                   x509v3-sign-rsa
x509v3-ecdsa-sha2-nistp521                   x509v3-sign-rsa-sha256@ssh.com
```

4.3.6 Cryptographic Hardware Support

Tectia Server for IBM z/OS can use the CP Assist for Cryptographic Functions (CPACF) and Cryptographic Coprocessors such as the CryptoExpress feature.

CPACF can be used to secure SSH network traffic with the AES algorithms for encryption (see Section 4.3.1) and the message authentication codes that are based on SHA-1 or SHA-2 (see Section 4.3.2). Note that the longer key lengths do not have CPACF support on all mainframe models.

The CPACF support for SHA-1 and SHA-2 is also used for digest calculations in key exchange and authentication.

The Tectia Server for IBM z/OS random number generator (RNG) can use cryptographic hardware support when adding entropy to its internal state. Tectia Server for IBM z/OS uses the ICSF Random Number Generate callable service if it is available (it requires a CryptoExpress feature). It will also use `/dev/random` if it is available.

Cryptographic hardware may be used in certificate-based authentication if the keys and certificates are stored in SAF and use RSA or ECC. Keys generated with the **RACDCERT** command can be stored in the Crypto-Express device or stored encrypted with a master key.

To use cryptographic hardware in Tectia Server for IBM z/OS the machine must be enabled for cryptography and the z/OS Integrated Cryptographic Service Facility (ICSF) must be active.

The configuration parameter **UseCryptoHardware** specifies how the cryptographic hardware is to be used. The value is a list of support values for algorithm groups and it may include a default support level. The support levels are:

- `no` - use the software implementation

- `yes` - use cryptographic hardware if available, otherwise software

- `must` - use cryptographic hardware, fail server startup if not available.

The algorithm groups are:

- `rng` - random number generator

- `sha` - SHA-1 and SHA-2 digest algorithms

- `aes` - AES algorithms

- `3des` - Triple DES

`sha1` may be used as a synonym of `sha`.

An example of the configuration parameters:

```
UseCryptoHardware yes,aes:must,sha:must
```

RACF users can control the use of the ICSF services with the CSFSERV class. If the class is defined, SSHD2, the user that runs the Tectia Server for IBM z/OS server, must have READ access to the CSFRNG profile if the random number generator support is to be used and to the CSFOWH profile if SHA support is to be used.

Enabling Use of IBM Crypto Express Card (CEX)

To enable cryptographic hardware you need to enable the CSFSERV profiles for all client, and server IDs in RACF. See Appendix I for instructions.

Ciphers AES-CBC, AES-CTR, AES-GCM, and 3DES-CBC, and Macs hmac-sha* are offloaded to CEX card, if they are configured in `sshd2_config`. CPACF will be used by default.

CEX related configuration parameters in `sshd2_config` are:

```
#      CryptoCardCipherIOThreshold    65536
Specifies the minimum size of cipher request that will be routed to
IBM cryptographic co-processor card (CEX), if the card is available and
UseCryptoHardware is set to yes/must, for cipher processing.  If the
request size is less than the CryptoCardCipherIOThreshold value, the
cipher request will be routed to CPACF facility.  Special values are
0 route all cipher requests to IBM cryptographic co-processor card
65536 or higher  route all cipher requests to CPACF facility

#      CryptoCardMACGenerate         no
Specifies whether to route MAC generation request to IBM cryptographic
co-processor card (CEX).  If it is set to yes, MAC generation request will
route to IBM cryptographic co-processor card (CEX), if the card is available
and UseCryptoHardware is set to yes/must, for MAC generation processing.
```

To use cryptographic hardware in Tectia Server for IBM z/OS the machine must be enabled for cryptography, and the z/OS Integrated Cryptographic Service Facility (ICSF) must be active. If IBM Crypto Express Card (CEX) is installed, Tectia Server for IBM z/OS will direct cipher operations to the co-processors in CEX via ICSF. The co-processors in CEX must be initialized with the master keys which are the same keys used to initialize the key data sets. You can refer to chapter *Using the pass phrase initialization utility in IBM z/OS ICSF Administrator's Guide* to initialize the co-processors in CEX.

> **ⓘ Note**
>
> There is a bug (APAR QA52113) in ICSF when processing AES cipher request on CEX card. We recommend you to install the related PTF when it is available. Toleration logic is implemented in v6.6 to bypass the deficiency.

4.3.7 Compressions

You can define the enabled compression methods for SSH connections in the `sshd2_config` file, using the `Compressions` keyword.

```
Compressions none,zlib
```

The available compression methods are:

- none

- zlib

To disable compression, set:

```
Compressions none
```

4.4 Configuring Root Logins

If you want to give someone permission to login directly to the root login account via Secure Shell, you can define three methods of control with the **PermitRootLogin** configuration parameter in the `sshd2_config` file:

The default value `yes` enables root logins with any authentication method:

```
PermitRootLogin      yes
```

Use the value `no` to disable all logins with root privileges:

```
PermitRootLogin      no
```

With the value `nopwd` root logins are allowed only when an authentication method other than password is used:

```
PermitRootLogin      nopwd
```

It is also possible to create a separate subconfiguration file for `root`. See Section 4.2 for more information.

4.5 Restricting User Logins

By default, Tectia Server for IBM z/OS does not impose any login restrictions in addition to those provided by the operating system. However, you can restrict connections based on host, user name, or group.

The restrictions are defined in the `sshd2_config` file using the following syntax:

```
keyword           pattern
```

> **Note**
>
> All the patterns used in the examples below are in accordance with the egrep syntax, which is the default regular expression syntax in Tectia Server for IBM z/OS.

Table 4.1. Examples of commonly used regular expressions and conventions with egrep syntax

Regex	Description
.*	matches everything
.	any character
\.	literal .
[:alpha:]+	any lower or uppercase alphabet character one or more times
(80|8080)	either 80 or 8080

The regex syntax can be chosen by using the metaconfig block in the beginning of `sshd2_config` and `ssh_certd_config` files:

```
## SSH CONFIGURATION FILE FORMAT VERSION 1.1
## REGEX-SYNTAX egrep
## end of metaconfig
```

Possible values of `REGEX-SYNTAX` are `ssh`, `egrep`, `zsh_fileglob` and `traditional`. For more information, see sshregex(1).

Previous versions of SSH Secure Shell (3.1 and earlier) always use the **zsh_fileglob** syntax.

Available keywords are the following:

- **AllowHosts** / **DenyHosts**

  Login is allowed/denied from hosts whose name matches one of the specified patterns.

  Example 1: Listing complete hostnames

  ```
  AllowHosts        localhost, example\.com, friendly\.example
  ```

  This allows connections only from specified hosts.

  Example 2: Using patterns with hostnames

```
AllowHosts            h..s.\..*
```

This pattern matches, for example, `house.foobar.com`, `house.com`, but not `house1.com`. Note that you have to input the string "`\.`" when you want to specify a literal dot.

Example 3: Using patterns with IP addresses

```
AllowHosts            ([[:digit:]]{1\,3}\.){3}[[:digit:]]{1\,3}
```

This pattern matches any IP address (`xxx.xxx.xxx.xxx`). However, some host's hostname could also match this pattern.

Example 4: Using `\i`

```
AllowHosts            "\i192.*\.3"
```

When `\i` is used in the beginning of a pattern, only the host IP addresses are used. The above pattern matches, for example, `192.0.0.3`.

- **AllowSHosts** / **DenySHosts**

  The `.shosts`, `.rhosts`, `/etc/shosts.equiv` and `/etc/hosts.equiv` entries are honored only for hosts whose name matches one of the specified patterns. It is recommended to use these keywords with host-based authentication.

- **AllowUsers** / **DenyUsers**

  Login is allowed/denied as users whose name matches one of the specified patterns.

  Example 1: Using complete user names

```
DenyUsers             devil@evil\.example,warezdude,1337
```

  This denies login as `devil` when the connection is coming from `evil.example`. It also denies login (from all addresses) as `warezdude` and as user whose UID is `1337`.

  Example 2: Using patterns with user names

```
AllowUsers            "sj*,s[:digit:]+,s(jl|amza)"
```

  This pattern matches, for example, `sjj`, `sjjj`, `s1`, `s123`, and `samza` but not `s1x` or `slj`.

  Example 3: Using `\i`

```
AllowUsers            "sjl@\i192.*\.3"
```

  This would allow login as user `sjl` from only those hosts whose IP address matches the specified pattern.

- **AllowGroups** / **DenyGroups**

  Login is allowed/denied when one of the groups the user belongs to matches one of the specified patterns.

Example 1

```
AllowGroups          root,staff,users
```

4.6 Configuring Code Pages

The code page and the line delimiter for terminal sessions can be configured with a series of configuration variables in the sshd2_config file. The variables have default values, which can be overridden with global, host-specific, or user-specific configurations.

The server supports the **chcp** command on interactive sessions. The command is typically used in the user's .profile logon script to set the code page to be applied in the terminal connections. For details, see the IBM document *z/OS USS User's Guide*.

```
ShellTransferCodeset         ISO8859-1
ShellTransferLineDelimiter   UNIX
ShellAccountCodeset          IBM-1047
ShellAccountLineDelimiter    UNIX
ShellTranslateTable          ""
ShellConvert                 yes
```

See also .

4.7 Defining Subsystems

Subsystems can be defined in the sshd2_config file using the following syntax:

```
subsystem-<name>      argument
```

The argument is the command which will be executed when the subsystem is requested.

```
$ sshg3 user@remote -s <name>
```

The argument can be a list of commands separated with a semicolon (;), or it can, for example, refer to a script.

One example of a subsystem is **sftp**.

```
subsystem-sftp       /opt/tectia/libexec/sft-server-g3
```

4.8 Auditing

Tectia Server for IBM z/OS logs events with syslog. Logging (auditing) is very important for security. You should check the logs often, or use tools to analyze them. From the logs, you can see whether unauthorized

access has been attempted, and take further action if needed. For example, you could add the hosts from which the attempts have been made to `DenyHosts`, or drop the packets from the domain completely at your firewall.

4.8.1 Configuring Logging in sshd2

sshd2 logs to the facility specified with the configuration option `SysLogFacility`. If the option is not set, **sshd2** logs to the AUTH facility.

For example, if you want **sshd2** to log to the LOCAL1 facility, you need to add the following setting to your server's configuration (`/opt/tectia/etc/sshd2_config`):

```
SysLogFacility        LOCAL1
```

You also need to modify syslog's configuration, so it knows where to put the log messages.

In `/etc/syslog.conf` (or equivalent):

```
local1.info          /var/log/sshd2
```

On some systems, this file may need to exist before syslog will write to it, so you may need to create it:

```
# touch /var/log/sshd2
```

If syslog accesses files with a non-root UID, for example logger, you need to change the ownership of the file to that user.

Remember to restart both **sshd2** and **syslogd** after making changes to their configuration files.

The following log facilities are available for SysLogFacility:

DAEMON	LOCAL0	LOCAL3	LOCAL6
USER	LOCAL1	LOCAL4	LOCAL7
AUTH	LOCAL2	LOCAL5	

4.8.2 Logging SFTP Transactions

The logging of SFTP transactions by the **sft-server-g3** subsystem can be controlled with the SftpSysLogFacility option. It accepts the same values as SysLogFacility (see Section 4.8.1). The default facility is DAEMON.

4.8.3 SMF Auditing

System Management Facilities (SMF) collect data for auditing. **sshd2** writes SMF records for failed login attempts. The **sft-server-g3** subsystem writes SMF records for the following events:

• Download a file (retrieve)

- Upload a file (store)

- Append data to a file

- Rename a file

- Delete a file

scpg3 and **sftpg3** clients write SMF records for the following events:

- Download to local file (store)

- Upload local file (retrieve)

The SMF record type for the **sshd2** server and the **sft-server-g3** subsystem can be defined with the **SftpSmfType** option in the server configuration (`/opt/tectia/etc/sshd2_config`):

```
SftpSmfType    TYPE119
```

For the **scpg3** and **sftpg3** clients the SMF record type can be defined in the `SSH_SFTP_SMF_TYPE` environment variable. The only available SMF record type is `TYPE119`.

Note that it is also possible to route syslog daemon messages to be stored in SMF record type 109. For details, see the IBM document *z/OS V1R6.0 CS: IP Configuration Reference, SC31-8776- 07*, chapter "Syslog daemon".

Note

If you intend to use OpenSSH SCP with Tectia Server for IBM z/OS, note that the default OpenSSH configuration on z/OS does not produce SMF records. SMF recording must be configured separately for OpenSSH when the OpenSSH SCP events need to be captured.

Required Permissions for SMF Records

The caller of the SMF service must be permitted to the `BPX.SMF` facility class profile:

- The `SSHD2` user must be permitted to the `BPX.SMF` facility class profile so that **sshd2** can create SMF records for users logging in and out.

- Each user that can transfer files must be permitted to the `BPX.SMF` facility class profile so that **sft-server-g3**, **scpg3**, and **sftpg3** can create SMF records for file transfers.

Give the following commands to set up the permissions:

```
RDEFINE FACILITY BPX.SMF UACC(NONE)
PERMIT BPX.SMF CLASS(FACILITY) ID(SSHD2) ACCESS(READ)
PERMIT BPX.SMF CLASS(FACILITY) ID(specific_username) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Changes in SMF TYPE119 Messages

All SMF records produced by **sshd2**, **sft-server-g3**, **scpg3**, and **sftpg3** are based on SMF type 119 record format described in the IBM document *z/OS V1R6.0 CS: IP Configuration Reference, SC31-8776-07*. Only subtypes 70 (FTP server transfer completion record), 72 (FTP server logon failure record), and 3 (FTP client transfer completion record) are used.

New values are used for `SMF119FT_FSLoginMech` in the FTP server security section and for `SMF119FT_FFLo-ginMech` in the FTP server login failure security section:

* `K (0xD2)` - public-key authentication

* `H (0xC8)` - host-based authentication.

In common TCP/IP identification section, new TCP/IP subcomponent values are used to distinguish the SFTP server and client from the FTP server and client. Value `SSHS` is used in **sshd2**, `SFTPS` is used in **sft-server-g3**, and `SFTPC` is used in file transfer clients **scpg3** and **sftpg3**.

4.9 Securing the Server

By default, the configuration of Tectia Client and Server is aimed toward usability. If you want to have a minimal, high-security configuration, you have to disable some functionality. This applies to both the client and the server.

This section presents four use case examples for restricting server configuration: restrictions to system admin-istration, file transfer and tunneling, and load control.

4.9.1 Restrictions to System Administration

Secure system administration is the traditional use case for Secure Shell.

Disabling Root Login

Usually, allowing direct root logins from the network is a bad idea. It is better to use forced commands to automate tasks requiring privileges (described in the section called "Forced Commands (with Public Keys)" below), and make people use **su** or **sudo** to elevate privileges otherwise.

In addition to `AllowUsers` and `DenyUsers`, you can easily disable root logins with passwords (see Section 4.4). Define the following in your `sshd2_config` file:

```
PermitRootLogin      nopwd
```

This way, jobs automated with forced commands will work.

Disabling Tunneling

If you are sure you or your users do not need to create tunnels (possibly going around firewall restrictions or such), you can disable tunneling (port forwarding) altogether by adding the following to your `sshd2_config` file:

```
AllowTcpForwarding       no
```

If you need more fine-grained control, consider using **`AllowTcpForwardingForUsers`** (and related keywords **`DenyTcpForwardingForUsers`**, **`AllowTcpForwardingForGroups`** and **`DenyTcpForwardingForGroups`**). With **`ForwardACL`** you can even allow and deny tunnels based on originator and destination (based on the IP address and port). See Section 4.9.3.

Disabling Terminal Access

If you only want to enable file transfers or tunneling for users in group `users`, you can disable terminal access with the configuration variable **`Terminal.DenyGroups`** in your `sshd2_config` file:

```
Terminal.DenyGroups      users
```

Other related keywords that can be used are: **`Terminal.AllowUsers`**, **`Terminal.DenyUsers`** and **`Terminal.AllowGroups`**.

It is recommended to deny also agent forwarding if terminal access is denied in `sshd2_config`:

```
AllowAgentForwarding     no
```

Note that the users will be able to use SFTP and other subsystems defined in the Tectia Server for IBM z/OS configuration. Any other "exec" and "shell" requests will be denied for the users. This includes forced commands with public keys described in the section called "Forced Commands (with Public Keys)" and the legacy style password changing when performed as forced command.

Forced Commands (with Public Keys)

If you have maintenance jobs requiring non-interactive access to your server, use public-key authentication and forced commands. This way, if the private key is compromised, the public key cannot be used to perform anything other than the predetermined command on the server. (This is, of course, also bad, but it would be worse if the malicious attacker would have unrestricted access to the machine.)

Do not use the root account for jobs where it is not absolutely necessary.

You can set up a forced command in the `authorization` file.

```
key backup-key.pub
options command="tar zxvf - /usr/local"
```

This would, on a successful login with `backup-key.pub`, force a backup job to start.

You can also use the command that was given on the **sshg3** command line:

```
key backup-key.pub
options command="echo $SSHG3_ORIGINAL_COMMAND"
```

Running **sshg3**:

```
% sshg3 localhost kukkuu
Authentication successful.
kukkuu
%
```

For more information on the public-key options in the authorization file, see Section 5.6.2.

Note that if the user or the user's group has been denied terminal access (with the **Terminal.DenyUsers** or **Terminal.DenyGroups** keywords), also forced commands will be denied.

Restricting Connections

Tectia Server for IBM z/OS can be configured to reject connection attempts from unknown hosts. For example the following allows connections only from the internal network `10.1.0.0/8` IP addresses and from an external host with the IP address `195.20.116.1`:

```
AllowHosts              \i10\.1\..*,\i195\.20\.116\.1
```

See Section 4.5 for information on the regular expression syntax and for more configuration examples and options.

On systems with several network interfaces, Tectia Server for IBM z/OS can also be bound to a specific network interface so that the server can be only accessed from the intended network. For example, the following will bind the listener to address `10.1.60.25` using port 2222:

```
ListenAddress       10.1.60.25
Port                2222
```

Logging

Tectia Server for IBM z/OS logs are most important for auditing. The logs also provide troubleshooting information, for example, when user authentication fails and a user is unable to log in. Please see Section 4.8 for more information.

Notification

It is recommended to notify the users before they decide to log in that their actions are logged. In some jurisdictions this is required.

To display for example the following text to the users before login, the banner message saved to `/opt/tectia/etc/ssh_banner_message` can be used:

```
Unauthorized use of this system is prohibited.
All actions are logged.
```

4.9.2 Restrictions to File Transfer

If Tectia Server for IBM z/OS is used for file transfer only, it is advisable to disable tunneling and terminal access to the server.

Enabling the SFTP Subsystem

To allow the users to connect with SFTP to Tectia Server for IBM z/OS, the secure file transfer subsystem has to be defined in the `sshd2_config` file:

```
subsystem-sftp   /opt/tectia/libexec/sft-server-g3
```

Restricting Access to User's MVS User Catalog

To restrict the users' access with SFTP, the `--attribute=zos-access` option can be specified with **sft-server-g3**:

```
subsystem-sftp   /opt/tectia/libexec/sft-server-g3 --attribute=zos-access:value
```

The values for the `zos-access` attribute are:

- `mvs`: User is allowed access only to the MVS side of the server.

- `usercatalog`: User is allowed access only to the MVS side of the server and only to his/her own catalog.

- `hfs`: User is allowed access only to the HFS side of the server.

- `mvs,hfs` or `hfs,mvs` or `all`: User is allowed access to both MVS and HFS sides of the server. No access restrictions are active.

- `usercatalog,hfs` or `hfs,usercatalog`: User is allowed access to the HFS side of the server and to the MVS side of the server but only to his/her own catalog.

The values are case-insensitive. You can use both "`MVS`" or "`mvs`". The values cannot include white spaces. Value "`mvs,hfs`" works, but "`mvs, hfs`" does not.

Example 1

To start **sft-server-g3** so that user can only access his/her own MVS catalog:

```
sft-server-g3 --attribute=zos-access:usercatalog
```

Example 2

To start **sft-server-g3** so that only HFS can be accessed:

```
sft-server-g3 --attribute=zos-access:hfs
```

Setting Up Security for Processing Offline Data Sets

Tectia Server for IBM z/OS can control who is permitted to request offline data sets to be mounted. Tape data sets are typically offline and a DASD data set is offline if the volume it resides on is not mounted. To control mounting, the System Authorization Facility (RACF, ACF2, or TSS) facility SSZ.MOUNT must be defined. When it is defined Tectia requires that the user has at least READ access before it requests a data set to be mounted. If SSZ.MOUNT is not defined there is no restriction on mounting.

The restriction is enforced both by the server and by the **sftpg3** and **scpg3** client programs.

In addition to having permission to request mounts, the user must also specify the file transfer attribute automount=yes for Tectia to allocate a data set with mounts allowed.

Tectia first attempts to allocate a data set without allowing the system to do a mount. If this fails because the data set is offline, and the user has the required permission, Tectia repeats the allocation and allows the system to mount the data set. The user can instruct Tectia to omit the first allocation by specifying automount=immed.

Note that a user who can open a shell or issue remote commands with Tectia has other ways of causing tape mount requests. To control tape mounts effectively, do the following steps:

1. Define SSZ.MOUNT with universal permission NONE:

    ```
    RDEFINE FACILITY (SSZ.MOUNT) UACC(NONE)
    ```

2. Set up the users who are to be allowed to mount tapes as file-transfer only users as instructed above (the section called "Restricting Access to User's MVS User Catalog ").

3. Give these users READ access to SSZ.MOUNT:

    ```
    PERMIT SSZ.MOUNT CLASS(FACILITY) ID(SRVACC1) ACCESS(READ)
    ```

4. After each command refresh the RACLIST:

    ```
    SETROPTS RACLIST(FACILITY) REFRESH
    ```

Disabling Tunneling

If you are sure you or your users do not need to create tunnels (possibly going around firewall restrictions or such), you can disable tunneling (port forwarding) altogether by adding the following to your sshd2_config:

```
AllowTcpForwarding      no
```

Note that this disables also transparent FTP tunneling through the server.

Disabling Terminal Access

The following configuration option of Tectia Server for IBM z/OS will deny the group sftpusers terminal access.

```
Terminal.DenyGroups        sftpusers
```

4.9.3 Restrictions to Tunneling

Tectia ConnectSecure can be used to transparently tunnel FTP traffic, as well as TN3270 connections from a Windows workstation, to the mainframe. Also other applications can be tunneled.

The following examples show how to restrict tunneling services for certain groups and how to deny terminal and file transfer services.

Please see Section 4.2 for information on user-specific configurations if more fine-grained control is needed over the services.

Restricting Tunneling

Transparent tunneling with ConnectSecure uses only local tunnels (port forwarding). The tunnels are established based on the configuration of the application being tunneled. Please see the *Tectia ConnectSecure Administrator Manual* for details on the tunneling principles.

The following configuration options of Tectia Server for IBM z/OS will deny remote port forwarding and allow local port forwarding for all users for example to `http://webserver.example.com` or `https://webserver.example.com`.

```
AllowTcpForwarding         yes
ForwardACL                 deny remote .* .*
ForwardACL                 allow local .* .*\.example\.com%(80|443)
```

The format for the value of the `ForwardACL` option is the following:

```
(allow|deny) (local|remote) user-pat forward-pat [originator-pat]
```

`user-pat` is used to match the client user, in the same way as in the `UserSpecificConfig` option.

With local port forwarding, `forward-pat` is a pattern of format `host-id[%port]`. `host-id` will match with the target host of the forwarding, in the same way as in the `AllowHosts` option. `port` will match the target port. If the client attempts to open the forwarding using a DNS name, the IP is looked up from the DNS, which will be used to match the pattern.

Note that the `ForwardACL` forward pattern defined with a DNS name does not match if the tunneled application uses IP addresses instead of DNS names for connections. The forward pattern defined with an IP address will match to both.

With local port forwarding, `originator-pat` will match the originator address that the client has reported. However, restrictions based on the source address of local port forwarding are normally not reliable because the client can forge the source address. `originator-pat` should be used only if the client can be trusted (for example, if it is administered by yourself).

If you specify any allow directives, all forwardings in that class (local or remote) not specifically allowed will be denied. If a forwarding matches both allow and deny directives, the forwarding will be denied.

Also, if you have specified any of the options {Allow,Deny}TcpForwardingFor{Users,Groups} or AllowTcpForwarding, and the forwarding for the user is disabled with those options, an allow directive will not re-enable the forwarding for the user.

The following example denies all forwarding for the sftpusers group. Other users are denied remote forwarding. User root is allowed all local forwarding. User tunnelu is allowed local forwarding only to the Telnet port (23) of addresses *.example.com, and the forwarding must originate from the client machine local address (127.0.0.1), it cannot be forwarded from a third host (this assumes that the client machines are trusted).

```
AllowTcpForwarding          yes
DenyTcpForwardingForGroups  sftpusers
ForwardACL                  deny remote .* .*
ForwardACL                  allow local root .*
ForwardACL                  allow local tunnelu .*\.example\.com%23 127.0.0.1
```

See Section 4.5 for more information on the egrep regular expression syntax used in configurations.

Note that the users with terminal (shell) access are restricted only in the Tectia Server for IBM z/OS configuration and can, for example, set up their own port forwardings. See Section 4.9.1 for more information.

Disabling Terminal Access

The following configuration option of Tectia Server for IBM z/OS will deny the user tunnelu terminal access.

```
Terminal.DenyUsers      tunnelu
```

Disabling File Transfers

To deny all users the access to the SFTP server, change the default SFTP subsystem configuration option of Tectia Server for IBM z/OS to:

```
subsystem-sftp
```

4.9.4 Load Control

The purpose of load control is to help keep Tectia Server for IBM z/OS running when the load is high (that is, the number of current connections is near the maximum allowed number of connections). High load might be caused by a connection flood denial-of-service attack that tries to make the server unavailable to its intended users by using so much of its resources that normal service is disrupted.

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

Load control uses four configuration variables in the `sshd2_config` file: `MaxConnections`, `LoadControl.Act-ive`, `LoadControl.DiscardLimit`, and `LoadControl.WhitelistSize`.

> **Note**
>
> It is recommended to set `LoadControl.Active` to `yes`, and `MaxConnections` to a proper value to prevent sshd2 from forking too many processes which might exceed the `USS BPXPRMxx MAXPROCSYS` value. It is also recommended to set `MaxConnections` to roughly half of the `MAXPROCSYS` value. Adjust the values of `TcpListenBacklog, TcpListenRate,` and `TcpListenPause` according to your CPU and network speeds.

The level of load is measured by how near the number of the server's current connections is to `MaxConnections`, the maximum number of connections that the server will handle simultaneously. The argument for `MaxCon-nections` is a positive number. The default value is `1000`, and the value `0` (zero) means that the number of connections is not limited. `MaxConnections` must be greater than 1 when load control is used.

`LoadControl.Active` can have a value of `yes` or `no`. The default value is `no` (load control is disabled). To enable load control, set `LoadControl.Active` to `yes`.

> **Note**
>
> If `MaxConnections` is set to `0` or `1`, load control is disabled even if you have set `LoadControl.Active` to `yes` in the `sshd2_config` file.

When the number of concurrent connections is greater than `LoadControl.DiscardLimit`, connections from IP addresses that have not recently had a successful authentication are discarded. When the number of concurrent connections is not greater than `LoadControl.DiscardLimit`, connections are accepted from any IP address (subject to restrictions defined with `AllowHosts` and `DenyHosts`). The allowed value range of `LoadCon-trol.DiscardLimit` is from 1 to `MaxConnections-1`. The default value is 90 percent of the value of `MaxCon-nections`. If you have not defined any configuration settings (that is, only `sshd2_config` default values are used), the value of `LoadControl.DiscardLimit` is `900`.

Tectia Server for IBM z/OS keeps a list of the IP addresses of connections that have had a successful authen-tication. This "white list" has space for a fixed number of unique IP addresses, specified by `LoadCon-trol.WhitelistSize`. The default value of `LoadControl.WhitelistSize` is 1000.

Chapter 5 Authentication

The Secure Shell protocol used by Tectia Server for IBM z/OS 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 Secure Shell server host can authenticate itself using either traditional public-key authentication or certificate authentication.

Different methods can be used to authenticate Secure Shell client users. These authentication methods can be combined or used separately, depending on the level of functionality and security you want.
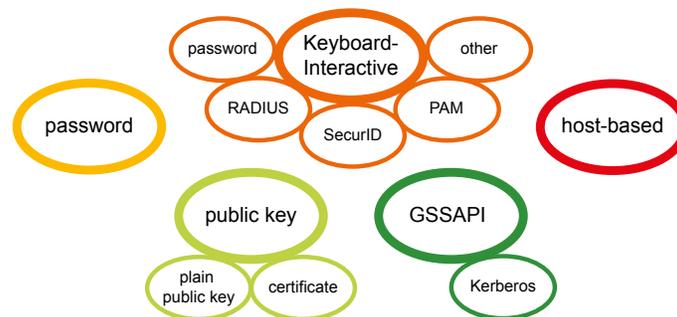
Figure 5.1. Secure Shell user authentication methods. Note that all of the methods are not available on z/OS.

Tectia Server for IBM z/OS allows public-key and password authentication by default. It also supports keyboard-interactive and host-based authentication.

5.1 Supported User Authentication Methods

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

Table 5.1. User authentication methods supported by the Tectia client/server solution

Authentication method	Tectia Server			Tectia Client, ConnectSecure, and client tools on Server for IBM z/OS		
	Unix[a]	Windows	z/OS	Unix[a]	Windows	z/OS
Password	x	x	x	x	x	x
Public-key	x	x	x	x	x	x
Certificate	x	x	x[b]	x	x	x[b]
Host-based	x	x	x	x		x
Keyboard-interactive	x	x	x	x	x	x
PAM[c]	x			x	x	x
RSA SecurID[c]	x	x		x	x	x
RADIUS[c]	x	x		x	x	x
GSSAPI/Kerberos	x	x		x	x	

[a] Including Tectia Server for Linux on IBM System z.

[b] Including certificates in files and SAF certificates.

[c] Through keyboard-interactive.

5.2 Using the z/OS System Authorization Facility

Tectia Server for IBM z/OS supports X.509 certificates and RSA/ECC keys managed by the z/OS System Authorization Facility (SAF). For more information, see Section 5.4.2 and Section 5.7.3. If SAF keys are going to be used, the users need permissions to access the relevant facilities.

Tectia Server for IBM z/OS uses the Integrated Cryptographic Services Facility (ICSF) if the `UseCryptoHardware` configuration variable allows it (see Section 4.3.6) or keys and certificates are stored in the System Authorization Facility (SAF) and the keys are of type ICSF or PCICC (see Section 5.4.2 and Section 5.7.3). SAF will control the use of cryptographic services if the `CSFSERV` class is activated and will control the access to cryptographic keys if the `CSFKEYS` class is activated.

When using SAF private keys, the server or client user needs access to the `CSFDSG` resource in the `CSFSERV` class. The private keys can be secured by permitting access to a resource in the `CSFKEYS` class. The name of the resource is the label of the key in the ICSF key database.

See the IBM document *z/OS ICSF Administrator's Guide*, chapter "Controlling Who Can Use Cryptographic Keys and Services" for instructions on how to use generic resource names, how to give permissions to user groups and connect users to groups, and how to define auditing.

The users (including the `SSHD2` user) must have at least READ access to the `IRR.DIGTCERT.LISTRING` facility. If a user needs access to a key ring belonging to another user, he must have UPDATE access to the facility. This case will arise when using a `KnownHostsEkProvider` for checking host certificates, because host certificates are best entered as SITE keys and are not owned by the verifier.

5.3 Server Authentication with Public Keys in File

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

5.3.1 Defining Server Host Key

The key pair used for server authentication is defined on the server in the `sshd2_config` file with the following parameters:

```
HostkeyFile             hostkey
PublicHostKeyFile       hostkey.pub
```

During the setup process, one RSA key pair (with the file names `hostkey` and `hostkey.pub`) is generated and stored in the `/opt/tectia/etc` directory. By default this key pair is used for server authentication. Make sure that only the user running **sshd2** has access to the private key.

In Tectia Server for IBM z/OS, each server daemon can have only one host key pair. This is different from Tectia Server on other platforms.

By default, the server uses a public key with the file name of the private key plus the extension `.pub`. The `PublicHostKeyFile` keyword has to be defined only if the public-key file is stored with a different file name.

5.3.2 Generating the Server Host Key Pair

The host public-key pair (2048-bit RSA) is generated during the installation of Tectia Server for IBM z/OS by running job KEYGH generated by Tectia SSH Assistant installation step 1.15 KEYGEN. You only need to regenerate the host key pair if you want to change it.

KEYGH invokes a tool called **ssh-keygen-g3** (located in `/opt/tectia/bin`) that generates the host key pair:

```
//KEYGH    EXEC SSZPBPX,BPX=BPXBATCH
//STDPARM  DD   *
SH /opt/tectia/bin/ssh-keygen-g3 -H ❶ -P ❷ -t rsa ❸
   -c "Tectia Server key for $(hostname) generated at $(date)" ❹
   -b 2048 ❺
//STDENV   DD   *
```

❶   The key pair will be stored in the default host key directory (`/opt/tectia/etc`).

❷   The key will be saved without a passphrase.

❸   The type of the key will be RSA.

❹   This line generates the key comment.

❺   The length of the key will be 2048 bits.

Because the key pair is generated in such a way that the private key has no passphrase (option -P), the server will start up without any operator interaction to enter a passphrase. Protect the key with file system access rules. The private key (/opt/tectia/etc/hostkey) must be accessible only by the SSHD2 user.

For more information on the key generation options, see the *Tectia Server for IBM z/OS User Manual* or the **ssh-keygen-g3** man page.

To (re)generate the host key in UNIX, perform the following tasks:

1. Use su to switch to a UID 0 user (if you are not already logged in as one).

2. Run **ssh-keygen-g3** to generate the host key, for example:

    ```
    # /opt/tectia/bin/ssh-keygen-g3 -t ecdsa -b 256 -P /opt/tectia/etc/hostkey
    ```

    This will generate a 256-bit ECDSA key pair without a passphrase and store it under /opt/tectia/etc.

3. Restart the server.

5.3.3 Using an OpenSSH Server Host Key

Tectia Server for IBM z/OS can use a key created with OpenSSH as the server host key. The key must be configured with the HostKeyFile option in sshd2_config or have the default file names, hostkey and hostkey.pub.

Both RSA and DSA keys of sizes from 512 to 4096 bits or more are supported, as well as ECC keys of sizes 256, 384, and 521 bits.

5.3.4 Notifying the Users of the Host Key Change

Administrators that have other users connecting to their server should notify the users of the host key change. If you do not, the users will receive a warning the next time they connect because the host key the users have saved on their disk for your server does not match the host key now being actually provided by your server. The users may not know how to respond to this error.

You can run the following to display a fingerprint of your new public host key which you can provide to your users via some unalterable method (for example, by a digitally signed e-mail or by displaying the fingerprint on a secured bulletin board):

```
$ /opt/tectia/bin/ssh-keygen-g3 -F hostkey.pub
```

When the users connect and receive the error message about the host key having changed, they can compare the fingerprint of the new key with the fingerprint you have provided to them, and ensure that they are connecting to the correct **sshd2** daemon. Inform your users that they should notify you if the fingerprints do not match, or if they receive a message about a host key change and do not receive a corresponding message from you notifying them of the change.

This procedure can help ensure that you do not become a victim of a man-in-the-middle attack, as your users will notify you if the host key fingerprints do not match. You will also be aware if the users encounter host key change messages when you have not regenerated your host key pair.

If you want to avoid the risk associated with the first connection, you can do one of the following:

- As an administrator of both the client and server machines, you can copy the server public key in advance to the global `hostkeys` directory on the client computer as `key_22_<hostname>.pub` (where `<hostname>` is the hostname the client uses when it connects to the server). The location of this directory depends on the operating system:

  - On Tectia client tools for z/OS: `/opt/tectia/etc/hostkeys`

  - On Tectia Client on Unix: `/etc/ssh2/hostkeys`

  - On Tectia Client on pre-Vista Windows: `"C:\Documents and Settings\All Users\Application Data\SSH\HostKeys"`

  - On Tectia Client on Windows Vista and later Windows versions: `"C:\ProgramData\SSH\HostKeys"`

  In this case, manual fingerprint check is not needed, and you can also set the `strict-host-key-checking` option in the `ssh-broker-config.xml` file on the client to `yes`. After this, Tectia Client will refuse to connect if the server's public key is not in the `hostkeys` directory.

- The server administrator can also send the public host key to the users via an unalterable method. The users can save the key in their `$HOME/.ssh2/hostkeys` directory as `key_22_<hostname>.pub`. If all remote host keys are received in this manner, the `strict-host-key-checking` option can be enabled on the client.

5.4 Server Authentication with Certificates

Tectia Server for IBM z/OS includes two implementations of certificate authentication. One is based on keys and X.509 certificates in files and software cryptography. This is the same implementation that is available in Tectia products on other platforms. The other implementation is based on keys and certificates managed by the z/OS System Authorization Facility (SAF) and cryptographic operations handled by the z/OS Integrated Cryptographic Service Facility (ICSF).

The two implementations can be combined or used separately. The Tectia validation can use trusted keys stored in file or in SAF, or the SAF validation can be used alone.

SAF Validation

ICSF is the interface to hardware cryptographic devices. Tectia Server for IBM z/OS benefits from the higher security and performance that these devices provide for keys and certificates managed by SAF.

Tectia Server for IBM z/OS also supports SAF keys that do not use hardware crypto devices, the so-called NON-ICSF keys. For these keys, the cryptographic operations are performed in software.

The interface to SAF in Tectia Server for IBM z/OS is implemented with a Tectia External Key Provider. The External Key Providers are configured with specification strings in a configuration file or on a command line.

Tectia Server for IBM z/OS validates public keys by matching them against trusted keys stored in the file system.

SAF does a limited form of certificate checking that only determines which SAF user is the owner of the certificate. SAF does not check the contents of the certificate, such as the validity period, or check for certificate revocation. Instead of revoking a certificate the site can reduce the user's access rights in SAF.

A trusted key provider must be configured if SAF certificate checking is to be used.

To enable SAF checking of remote Secure Shell servers, their certificates can be entered into SAF as SITE keys and placed on a key ring for the trusted key provider.

Tectia Certificate Validation

The Tectia Certificate Validator does a full validation of the certificate and can be configured to use external PKI services such as LDAP servers that store revocation information.

When a trusted key provider is configured, the Tectia validator takes its trusted CA certificates from SAF, otherwise they are read from files.

Tectia Server for IBM z/OS can be configured to support either public-key or certificate authentication. With certificate authentication, the private key and certificate can be stored either in SAF or in file.

It is also possible to configure the server to use a key from SAF and use only the public key extracted from the certificate for authentication.

If a SAF key is configured but the key cannot be found or ICSF is required but not available, the server will issue an error message and will not start up.

5.4.1 Certificates Stored in File

To configure Tectia Server for IBM z/OS to authenticate itself using X.509 certificates from file, perform the following tasks:

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

    Note that the DNS address extension (`dns`) in the certificate needs to correspond to the fully qualified domain name of the server.

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

```
# ssh-cmpclient-g3 INITIALIZE \
   -p 62154:secret \
   -P generate://ssh2@rsa:2048/testserv-rsa \
   -s "C=FI,O=SSH,CN=testserv;dns=testserv.ssh.com" \
   -o /opt/tectia/etc/testserv-rsa \
   -S http://fw.example.com:1080 \
   http://pki.example.com:8080/pkix/ \
   'C=FI, O=SSH, CN=Test CA 1'
```

For more information on **ssh-cmpclient-g3** and **ssh-scepclient-g3**, see their man pages.

2. Define the private key and the server certificate in the `/opt/tectia/etc/sshd2_config` file, for example, using the key and certificate created above:

```
HostKeyFile             testserv-rsa.prv
HostCertificateFile     testserv-rsa-0.crt
HostKey.Cert.Required    no
```

Setting the `sshd2_config` option **HostKey.Cert.Required** to `yes` defines that the server must authenticate with a certificate. When keys in file are used, a certificate must be defined with the **HostCertificateFile** option. Setting the option to `no` (default) means that the server can use either a normal public key or a certificate, depending on which of them is configured. Setting the option to `optional` means that the server can use both a certificate and the public key found in the certificate.

3. Restart the server as instructed in Section 3.1.3.

5.4.2 Certificates Stored in SAF

The following example assumes that the SSHD2 user created in Section 2.3.4 is used to run the server.

To use SAF certificates for authenticating the server, do the following steps. Replace the names and IDs with those appropriate to your system:

1. Create the server host key in SAF by giving the following TSO commands:

```
RACDCERT ID(SSHD2) GENCERT SUBJECTSDN(CN('LPAR1') OU('RD')
  O('EXAMPLE')) SIZE(2048) WITHLABEL('LPAR1.EXAMPLE.COM')
RACDCERT ID(SSHD2) LIST
```

> ℹ **Note**
>
> The above command will create a 2048-bit RSA key. If you want to, for example, create a 521-bit ECC key, replace `SIZE(2048)` with:
>
> ```
> SIZE(521) NISTECC
> ```

2. Give the following TSO command to generate the certification request:

```
RACDCERT ID(SSHD2) GENREQ(LABEL('LPAR1.EXAMPLE.COM'))
  DSN('SSHD2.LPAR1.CRT.REQ')
```

3. Use the PKCS#10 certification request in the data set `'SSHD2.LPAR1.CRT.REQ'` to enroll the certificate. The actual steps depend on your CA setup.

4. After the enrollment is completed, store the received certificate to a data set, for example `'SSHD2.LPAR1.CRT'`.

5. To connect the new certificate to a key ring, give the following TSO commands:

```
RACDCERT ID(SSHD2) ADD('SSHD2.LPAR1.CRT') TRUST
  WITHLABEL('LPAR1.EXAMPLE.COM')
RACDCERT ID(SSHD2) ADDRING(SSH-HOSTKEY)
RACDCERT ID(SSHD2) CONNECT(ID(SSHD2) LABEL('LPAR1.EXAMPLE.COM')
  RING(SSH-HOSTKEY) USAGE(PERSONAL))
RACDCERT ID(SSHD2) LISTRING(SSH-HOSTKEY)
```

6. For the settings to take effect, give the following TSO command:

```
SETROPTS RACLIST(DIGTCERT) REFRESH
```

7. Define the z/OS SAF external key provider in the `/opt/tectia/etc/sshd2_config` file:

```
HostKeyEkProvider        zos-saf
HostKeyEkInitString      "KEYS(ID(SSHD2) RING(SSH-HOSTKEY)
  LABEL('LPAR1.EXAMPLE.COM'))"
HostKey.Cert.Required    yes
```

Note that **HostKeyEkInitString** must point to a single private key. Setting **HostKey.Cert.Required** to `yes` defines that the server must authenticate with a certificate. When the z/OS SAF provider is used, setting the option to `no` means that only the public key found in the SAF certificate is used. Setting the option to `optional` means that both the SAF certificate and the public key found in the SAF certificate are used.

For more information on the configuration file options, see sshd2_config(5). For information on the format of the external key initialization string, see ssh-externalkeys(5).

5.5 User Authentication with Passwords

The password authentication method is the easiest to implement, as it is enabled by default. Password authentication uses the RACF system password of the user.

To allow password authentication, make sure that the **AllowedAuthentications** keyword of the `/opt/tectia/etc/sshd2_config` configuration file on the server contains the argument `password`:

```
AllowedAuthentications        password
```

Other authentication methods can be listed in the configuration file as well.

To allow the users to change expired passwords, uncomment the following line in the `/opt/tec-tia/etc/sshd2_config` file:

```
AuthPassword.ChangePlugin       ssh-passwd-plugin
```

> **Note**
>
> With passwords, it is also possible to use keyboard-interactive authentication. See Section 5.9 for more information.

5.6 User Authentication with Public Keys in File

Public-key authentication is based on the use of digital signatures and provides very good authentication security. To use public-key authentication, the user must first create a key pair on the client, and upload the public key to the server.

5.6.1 Enabling Public-Key Authentication

To enable user public-key authentication on Tectia Server for IBM z/OS (it is allowed by default), make sure the **AllowedAuthentications** keyword in the `/opt/tectia/etc/sshd2_config` file contains the argument `publickey`:

```
AllowedAuthentications  publickey
```

Other authentication methods can be listed in the configuration file as well.

5.6.2 Using the Authorization File

Tectia Server for IBM z/OS requires an authorization file that lists the user public keys that are authorized for login.

The default location for the authorization file is `$HOME/.ssh2/authorization`. The file location can be changed with the **AuthorizationFile** keyword in the `sshd2_config` file.

The authorization file contains a list of public key file names each preceded by the keyword `Key`. If there is more than one `Key`, they are all authorized for login.

Authorization File Options

It is possible to define different settings in the authorization file depending on which key is used in public-key authentication. The authorization file has the same general syntax as the `sshd2_config` configuration file. The following keywords may be used:

Key

> This is followed by the file name of a public key in the user configuration directory (by default, $HOME/.ssh2) that is used for identification when contacting the host. If there is more than one key defined, they are all acceptable for login.

Options

> This keyword, if used, must follow the Key keyword above. The various options are specified as a comma-separated list.
>
> The following Options can be used:

allow-from and **deny-from**

> In addition to public-key authentication, the canonical name of the remote host must match the given pattern(s). These parameters follow the logic of the **AllowHosts** and **DenyHosts** keywords of sshd2_config. Specify one pattern per keyword; multiple keywords can be used.

command="command"

> This is used to specify a "forced command" that will be executed on the server side instead of anything else when the user is authenticated. The command supplied by the user (if any) is put in the environment variable SSH2_ORIGINAL_COMMAND. The command is run on a pty if the connection requests a pty; otherwise it is run without a tty. Quotes may be used in the command if escaped with backslashes.
>
> This option might be useful for restricting certain public keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Note that the client may specify TCP/IP and/or X11 forwarding, unless they are explicitly prohibited (see no-port-forwarding).

environment="NAME=value"

> Specifies that the string is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted.

idle-timeout=time

> Sets idle timeout limit to time either in seconds (s or nothing after the number), in minutes (m), in hours (h), in days (d), or in weeks (w). If the connection has been idle (all channels) this long, the connection is closed.

no-port-forwarding

> Forbids TCP/IP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This is useful in combination with the command option.

no-agent-forwarding

> Forbids authentication agent forwarding when this key is used for authentication.

no-pty

> Prevents tty allocation (a request to allocate a pty will fail).

Example: Your `authorization` file could, for example, contain the following:

```
Key master.pub
Key maid.pub
Options allow-from=".*\.example\.org"
Key butler.pub
Options deny-from=".*\.evil\.example",no-pty
```

When someone now logs in using the `master` key, the connection is not limited in any way by the `authorization` file. However, if the `maid` key is used, only connections from certain hosts will be allowed. And if the `butler` key is used, connections are denied from certain hosts, and additionally the allocation of tty is prevented.

5.6.3 Using Keys Generated with OpenSSH

If the user has an existing OpenSSH `authorized_keys` file on the server, the **ssh-keygen-g3** tool can be used to import the OpenSSH `authorized_keys` file and to configure the `authorization` file, for example:

```
SERVER> ssh-keygen-g3 --import-ssh1-authorized-key
$HOME/.ssh/authorized_keys $HOME/.ssh2/authorization

Imported key /home/user/.ssh/authorized_keys:1 to
/home/user/.ssh2/imported-437b1a07-1.pub and added to authorization file
/home/user/.ssh2/authorization
Imported key /home/user/.ssh/authorized_keys:2 to
/home/user/.ssh2/imported-437b1a07-2.pub and added to authorization file
/home/user/.ssh2/authorization
```

For more information on the **ssh-keygen-g3** options, see the *Tectia Server for IBM z/OS User Manual* or the **ssh-keygen-g3** man page.

Alternatively, the administrator of Tectia Server for IBM z/OS may enable **AuthorizedKeysFile** in the server configuration file `/opt/tectia/etc/sshd2_config`, for example as follows:

```
AuthorizedKeysFile %D/.ssh/authorized_keys
```

Tectia Server for IBM z/OS will check the defined `AuthorizedKeysFile` in addition to the user's `AuthorizationFile` (by default `$HOME/.ssh2/authorization`). Note that the `AuthorizationFile` has precedence over `AuthorizedKeysFile` if the same key is defined in both.

5.7 User Authentication with Certificates

Tectia Server for IBM z/OS includes two implementations of certificate authentication. One is based on keys and X.509 certificates in files and software cryptography. This is the same implementation that is available in Tectia products on other platforms. The other implementation is based on keys and certificates managed by the z/OS System Authorization Facility (SAF) and cryptographic operations handled by the z/OS Integrated Cryptographic Service Facility (ICSF).

For more information, see Section 5.4.

The server can be configured to allow or require certificate-based user authentication. To use SAF certificates, a trusted key provider must be configured. The users must be set up with digital certificates.

When using a certificate, the client can start authentication without presenting a user name. If the user name given by the user matches the value of the `IdentityDispatchUsers` keyword in the server configuration, the name retrieved from SAF will be used. However, it is not allowed to change the user ID during the authentication process. For example, if the server requires first certificate authentication and then password authentication, the user must give the password for the user that SAF determines from the certificate.

SAF determines the z/OS user name using one-to-one certificate to user ID association, certificate name filtering, or the **HostIdMappings** certificate extension. Tectia Server for IBM z/OS does not participate in this processing.

The server checks the user certificate using SAF and can be configured to do a full PKI validation using the Tectia Certificate Validator.

5.7.1 Certificates Stored in File

To configure the server to allow user authentication with certificates, perform the following tasks:

1. Acquire the CA certificate and copy it to the server machine. You can either copy the X.509 certificate(s) as such or you can copy a PKCS #7 package including the CA certificate(s). Certificates can be extracted from a PKCS #7 package by specifying the `-7` option with **ssh-keygen-g3**.

2. Certificate authentication is a part of the `publickey` authentication method. Make sure that you have enabled it in the `/opt/tectia/etc/sshd2_config` file:

```
AllowedAuthentications        publickey
AuthPublicKey.Cert.Required   no
```

Setting the `AuthPublicKey.Cert.Required` option to `yes` defines that the user must authenticate with a certificate or else the authentication will fail.

3. Specify the trusted CA certificate and the mapping file(s) in the `ssh_certd_config` file:

```
Pki                       <ca-cert-path>
MapFile                   <map-file-path>
```

You can define several CA certificates by using several `Pki` keywords.

```
Pki                       test-ca1.crt
MapFile                   cert-user-mapping1.txt
Pki                       test-ca2.crt
MapFile                   cert-user-mapping2a.txt
MapFile                   cert-user-mapping2b.txt
```

Note that multiple **`MapFile`** keywords are permitted per `Pki` keyword. Also, if no mapping file is defined, all connections are denied even if user certificates can be verified using the defined CA certificate. The server will accept only certificates issued by defined CA(s).

4. Also define the LDAP server(s) used for CRL checks in the `ssh_certd_config` file. If the CA services (OCSP, CRLs) are located behind a firewall, define also the SOCKS server.

```
LdapServers                  ldap://ldap.example.com:389
SocksServer                  socks://fw.example.com:1080
```

Defining the LDAP server is not necessary if the CA certificate contains a `CRL Distribution Point` or an `Authority Info Access` extension.

5. Create the certificate user mapping file as described in Section 5.7.2.

6. Restart **ssh-certd** as instructed in Section 3.2.3.

For more information on the configuration file options, see sshd2_config(5) and ssh_certd_config(5).

5.7.2 Certificate User Mapping File

The map file specifies which certificates authorize logging into which accounts. The format of the file is as follows:

```
<account-id> <keyword> <argument>
```

The `keyword` can be either `Email`, `Subject`, `SerialAndIssuer`, `EmailRegex`, or `SubjectRegex`. The `argument` depends on the `keyword`.

- `Email`: The argument is the e-mail address which must be present in the certificate.

- `Subject`: The argument is the required subject name in LDAP DN (distinguished name) string format.

- `SerialAndIssuer`: The argument is the required serial number and issuer name in LDAP DN string format, separated by spaces or tabs.

- `EmailRegex`: The argument is the regular expression which must match an e-mail address in the certificate. If `account-id` contains the string `%subst%`, it is substituted with the first parenthesized part of the regular expression. The patterns are matched using the egrep syntax.

- `SubjectRegex`: The argument is the regular expression which must match a subject name in the certificate. If `account-id` contains the string `%subst%`, it is substituted with the first parenthesized part of the regular expression. The patterns are matched using the egrep syntax.

Examples

The following are examples of different map file definitions:

```
user1 email user1@ssh.com
user1 subject C=FI,O=SSH,CN=Secure Shell User 1
user1 serialandissuer 1234 C=FI,O=SSH,CN=Secure Shell User 1
%subst% subjectregex C=FI, O=SSH, CN=([a-z]+)
%subst% emailregex ([a-z]+)@ssh\.com
```

The last line permits logging with any e-mail address with only letters in the user name. For more information on the regular expression syntax, see sshregex(1).

5.7.3 Certificates Stored in SAF

To configure the server to allow user authentication with SAF certificates, perform the following tasks. Replace the names and IDs with those appropriate to your system:

1. Certificate authentication is a part of the `publickey` authentication method. Make sure that you have enabled it in the `/opt/tectia/etc/sshd2_config` file:

    ```
    AllowedAuthentications                  publickey
    AuthPublicKey.Cert.Required             yes
    ```

    Setting the **AuthPublicKey.Cert.Required** option to `yes` defines that the user must authenticate with a certificate or else the authentication will fail.

2. Define the certificate validation method in the `/opt/tectia/etc/sshd2_config` file. The validation method can be either `saf`, `tectia`, or both (`saf,tectia`).

    ```
    AuthPublicKey.Cert.ValidationMethods    tectia|saf|saf,tectia
    ```

    If `saf` is specified, RACF/SAF is used for validating user certificates. The user certificates must be found in a trusted key ring defined by the **AuthorizationEkProvider** keyword. Note that when only SAF validation is used, the certificate validity period and revocation status are not checked.

    If `tectia` is specified (or the keyword is missing from the configuration), the Tectia Certificate Validator (**ssh-certd**) is used for validating the user certificate. The user certificates must be issued by a trusted certification authority defined in the **Pki** or **PkiEkProvider** keyword of `ssh_certd_config`.

    If both values are specified, RACF/SAF determines the user name based on the certificate contents. After that the Tectia validation is performed. The CA certificate of the issuing certification authority must be found in a trusted key ring defined by the `PkiEkProvider` keyword of `ssh_certd_config`.

    > ## Note
    >
    > Several of the following steps are marked either as *SAF validation*, *SAF validation only*, *Tectia validation*, or *Tectia validation only*. These steps are alternatives depending on the value of **AuthPublicKey.Cert.ValidationMethods**. If `saf,tectia` is selected, both the steps marked as *SAF validation* and *Tectia validation* need to be completed.

3. (*SAF validation only*) Get the user certificate and store it to a data set, for example `'USER.CRT'`.

4. (*SAF validation only*) To add the user certificate into SAF, give the following TSO commands:

```
RACDCERT ID(USER) ADD('USER.CRT')TRUST WITHLABEL('USER')
RACDCERT ID(USER) ADDRING(USER)
RACDCERT ID(USER) CONNECT(ID(USER) LABEL('USER') RING(USER)
  USAGE(PERSONAL))
RACDCERT ID(USER) LISTRING(USER)
```

5. (*SAF validation only*) For the settings to take effect, give the following TSO command:

```
SETROPTS RACLIST(DIGTCERT) REFRESH
```

6. (*Tectia validation*) Get the CA certificate and store it to a data set, for example `'PKICA.CRT'`.

7. (*Tectia validation*) To add the CA certificate into SAF, give the following TSO commands:

```
RACDCERT CERTAUTH ADD('PKICA.CRT') TRUST WITHLABEL('PKICA')
RACDCERT ID(SSHD2) ADDRING(SSH-PKI)
RACDCERT ID(SSHD2) CONNECT(CERTAUTH LABEL('PKICA') RING(SSH-PKI)
  USAGE(CERTAUTH))
RACDCERT ID(SSHD2) LISTRING(SSH-PKI)
```

8. (*Tectia validation*) For the settings to take effect, give the following TSO command:

```
SETROPTS RACLIST(DIGTCERT) REFRESH
```

9. (*SAF validation*) The **`IdentityDispatchUsers`** keyword in the `/opt/tectia/etc/sshd2_config` file can be used to define user name patterns for which the login name given by the user is not used but the real user name is taken from the user certificate.

```
IdentityDispatchUsers          username-pattern
```

10. (*SAF validation*) A user certificate might contain a **HostIdMapping** extension field. That field is used by SAF to determine the local user name of the user. If the user certificate has the correct host name in the **HostIdMapping** host name field, the user name associated with that host name (specified in the certificate) is used. To use the **HostIdMapping** extension in user certificates, give the CA certificate the `HIGHTRUST` status and give the `SSHD2` user READ access to the resource `IRR.HOST.hostname` in the `SERVAUTH` class, where `hostname` is the character string used in the certificate extension. Often the DNS name of the server is used as the host name. To allow SAF to use the **HostIdMapping** extension, give the following commands:

```
RACDCERT CERTAUTH ALTER(LABEL('PKICA')) HIGHTRUST
RDEFINE SERVAUTH IRR.HOST.LPAR2.EXAMPLE.COM UACC(NONE)
PERMIT IRR.HOST.LPAR2.EXAMPLE.COM CLASS(SERVAUTH) ID(SSHD2) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
RLIST SERVAUTH IRR.HOST.LPAR2.EXAMPLE.COM ALL
```

11. (*SAF and Tectia validation*) As an alternative to the previous step, when both SAF and Tectia validation are used, the **HostIdMappingHostnames** keyword in the /opt/tectia/etc/sshd2_config file can be used to define a list of host names that the server recognizes. If the user certificate has a matching host name in the **HostIdMapping** host name field, the user name associated with that host name (specified in the certificate) is used.

```
HostIdMappingHostnames        lpar2.example.com
```

12. (*SAF validation only*) If only SAF validation is used, define the z/OS SAF external key provider that contains the user certificates with the **AuthorizationEkProvider** keyword in the /opt/tectia/etc/sshd2_config file:

```
AuthorizationEkProvider       "zos-saf:KEYS(ID(%UU) RING(%UU))"
```

The AuthorizationEkProvider keyword can contain special strings in the key specification that are mapped according to the following list:

- %U = user name

- %UU = user name in upper case

- %UL = user name in lower case

- %IU = user ID

- %IG = user group ID

13. (*Tectia validation*) If Tectia validation is used, define the z/OS SAF external key provider that contains the CA certificates with the **PkiEkProvider** keyword in the ssh_certd_config file:

```
PkiEkProvider                 "zos-saf:KEYS(ID(SSHD2) RING(SSH-PKI))"
```

14. (*Tectia validation only*) If only Tectia validation is used, define also the mapping file(s) in the ssh_certd_config file:

```
PkiEkProvider                 "zos-saf:KEYS(ID(SSHD2)RING(SSH-PKI))"
MapFile                       cert-user-mapping.txt
```

15. (*Tectia validation*) If Tectia validation is used, define also the LDAP server(s) used for CRL checks in the ssh_certd_config file. If the CA services (OCSP, CRLs) are located behind a firewall, define also the SOCKS server.

```
LdapServers                   ldap://ldap.example.com:389
SocksServer                   socks://fw.example.com:1080
```

Defining the LDAP server is not necessary if the CA certificate contains a CRL Distribution Point or an Authority Info Access extension.

---

16. (*Tectia validation only*) If only Tectia validation is used, create the certificate user mapping file as described in Section 5.7.2.

17. (*Tectia validation*) Restart **ssh-certd** as instructed in Section 3.2.3.

For more information on the configuration file options, see sshd2_config(5) and ssh_certd_config(5). For information on the format of the external key initialization string, see ssh-externalkeys(5).

5.8 Host-Based User Authentication

Host-based authentication uses the public host key of the client machine to authenticate a user to the remote server daemon (**sshd2**). This provides a non-interactive form of authentication, and is best used in scripts and automated processes, such as cron jobs. Host-based authentication can be used to automate backups and file transfers, or in other situations where a user will not be present to input authentication information.

The nature of any non-interactive login is inherently insecure. Whenever authentication without user challenge is permitted, some level of risk must be assumed. If feasible, public-key authentication is preferred. Tectia Server for IBM z/OS provides host-based authentication as a form of non-interactive login that is more secure than the `.rhosts` method used by the Berkeley 'r' commands, but it cannot resolve the inherent insecurity of non-interactive logins.

This means that you should take aggressive measures to ensure that any client machine set up for host-based authentication is adequately secured, both by software and hardware, to prevent unauthorized logins to your server.

Setting up host-based authentication requires administrator rights on the client machine. Both ends of the configuration are explained in this manual.

In the following instructions, `Server` is the remote host running Tectia Server for IBM z/OS to which you are trying to connect. `ServerUser` is the user name on `Server` that you are logging in as. `Client` is the host running Tectia client tools for z/OS. `ClientUser` is the user name on `Client` that should be allowed to log in to `Server` as `ServerUser`.

5.8.1 Client Configuration

When a user connects to a server using host-based authentication, it must prove that it has proper client host credentials (host private key, host public key, and/or host certificate). As the client itself (**sshg3**) cannot access the host private key, there is a separate binary (**ssh-signer2**) that has sufficient permissions for private key operations. **ssh-signer2** reads the default server configuration file, `/opt/tectia/etc/sshd2_config`.

Host-based authentication can be enabled either by using traditional public keys or by using certificates. In Tectia Server for IBM z/OS, the host public key and/or certificate must be stored in software (**ssh-signer2** cannot access it from SAF).

Traditional Public Keys Stored in File

To enable host-based authentication with traditional public keys on Tectia client tools for z/OS, perform the following steps as `ClientUser`:

1. Generate a host key. By default, `/opt/tectia/etc/hostkey` and `/opt/tectia/etc/hostkey.pub` are generated during installation, so you can skip this step. Otherwise, give the following command as a UID 0 user:

```
# /opt/tectia/bin/ssh-keygen-g3 -P /opt/tectia/etc/hostkey
```

2. Add the following line in the `ssh-broker-config.xml` file:

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

Also other authentication methods can be listed. Place the least interactive method first (this usually means the host-based method).

3. Change the `DefaultDomain` element in the `ssh2_config` file to reflect your fully qualified domain:

```
DefaultDomain     .example.com
```

> **Note**
>
> On Tectia Server 6.6 for IBM z/OS, the `ssh2_config` file is used solely for host-based authentication and it does not exist by default. You have to create it in `/opt/tectia/etc/ssh2_config` or `$HOME/.ssh2/ssh2_config`. Setting this is mandatory if the **HostbasedAuthForceClientHostnameDNSMatch** keyword in the `sshd2_config` file on `Server` has been set to `yes`. But even if `HostbasedAuthForceClientHostnameDNSMatch` is not used, the `DefaultDomain` keyword is useful on systems that report only the short host name by default.

Certificates Stored in File

It is possible to use a certificate instead of the traditional public-key pair to authenticate the client host. In Tectia Server for IBM z/OS, the host certificate must be stored in software (**ssh-signer2** cannot access it from SAF).

To enable host-based authentication with certificates on Tectia client tools for z/OS, do the following steps as `ClientUser`:

1. Add the following line in the `ssh-broker-config.xml` file:

```
<authentication-methods>
  <auth-hostbased />
```

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

2. Enroll a certificate for `Client`. See Section 5.7 for more information. The certificate must contain a `dns` extension which contains the fully qualified domain name (FQDN) of `Client`. Note that the private key associated with the certificate needs to be stored with an empty passphrase.

3. Define the private key and certificate in `sshd2_config` on `Client`:

```
HostKeyFile              <private key>
HostCertificateFile      <server-certificate>
```

4. Change the `DefaultDomain` element in the `ssh2_config` file to reflect your fully qualified domain:

```
DefaultDomain            .example.com
```

> ### Note
>
> On Tectia Server 6.6 for IBM z/OS, the `ssh2_config` file is used solely for host-based authentication and it does not exist by default. You have to create it in `/opt/tectia/etc/ssh2_config` or `$HOME/.ssh2/ssh2_config`.

5.8.2 Server Configuration

Host-based authentication can be enabled either by using traditional public keys or by using certificates.

Traditional Public Keys Stored in File

To allow host-based authentication with traditional public keys on Tectia Server for IBM z/OS, perform the following steps as `ServerUser`:

1. Create a file named `.shosts` in the home directory of `ServerUser`. The contents of this file should be the client's host name, some tabs or spaces, and the user's user name on the client.

```
client.example.com      ClientUser
```

Make sure the `.shosts` file is owned by `ServerUser` and its permissions are `0600`.

2. Check that the server user's home directory is owned by the user and its permissions are at most `0755` (or more restrictive, like `0700`).

If every user is allowed to write to the directory, there will be nothing to prevent them from overwriting the `.shosts` file with their own version with an entry for their client user, allowing them to authenticate to Tectia Server as `ServerUser`.

Do the following as the server administrator:

1. Copy the client's `hostkey.pub` file over to the server. Note that this requires root permissions on the client, and optionally on the server as well.

   Tectia Server for IBM z/OS is configured by default to look in one of two places on server for the host keys to use for host-based authentication:

   ```
   /opt/tectia/etc/knownhosts
   ```

   OR

   ```
   $HOME/.ssh2/knownhosts
   ```

   The server administrator can edit the **UserKnownHosts** keyword in the `/opt/tectia/etc/sshd2_config` file to disable the use of the user-defined known hosts (they are allowed by default).

   If you want to allow host-based authentication to all users connecting from the client machine, you can add the public host key to `/opt/tectia/etc/knownhosts`. Root permissions are required for this method.

   If you want to allow host-based authentication only to some users, and if user-defined known hosts are allowed, then you can instead add the keys to the `$HOME/.ssh2/knownhosts` directory.

   You have to name the client's public key as follows on the server:

   ```
   client.example.com.ssh-rsa.pub
   ```

   In the example, `client.example.com` is the host name the client is sending to the server. When `Default-Domain` has been set on client, this name is always the long host name (FQDN). This gives the server the client's public key so the server can verify the client user's identity based on the public key signature.

2. In the `/opt/tectia/etc/sshd2_config` file under the **AllowedAuthentications** keyword, add `hostbased` as an allowed method. For example:

   ```
   AllowedAuthentications        hostbased,publickey,password
   ```

   If you want to allow multiple authentication methods, place the least interactive method first.

3. Restart the server as instructed in Section 3.1.3.

For more information on the configuration file options, see sshd2_config(5).

To test that host-based authentication works, log in to `Client` as `ClientUser` and run the following command:

```
$ sshg3 ServerUser@Server uptime
```

You should get back the results of uptime on the server.

Certificates Stored in File

Tectia Server for IBM z/OS checks the host certificate using SAF and can be configured to do a full PKI validation using the Tectia Certificate Validator (see Section 5.4). The certificates of the remote hosts can be

entered into SAF as SITE certificates and connected to the key ring configured for host authentication, or they can be issued by a CA that has its CA certificate on the server's key ring.

To allow host-based authentication with certificates on the server, do the following steps as `ServerUser`:

1. Create a file named `.shosts` in the home directory of `ServerUser`. The contents of this file should be the client's host name, some tabs or spaces, and the user's user name on the client.

```
client.example.com        ClientUser
```

Do the following steps as the server administrator:

1. In the `/opt/tectia/etc/sshd2_config` file under the **AllowedAuthentications** keyword, add `hostbased` as an allowed method. For example:

```
AllowedAuthentications          hostbased,publickey,password
AuthHostbased.Cert.Required     no
```

   If you want to allow multiple authentication methods, place the least interactive method first. Setting the **AuthHostbased.Cert.Required** option to `yes` defines that the client host must authenticate with a certificate or else the authentication will fail.

2. Specify the trusted CA certificate in the `ssh_certd_config` file:

```
HostCA                          trusted-ca.crt
```

3. Also define the LDAP server(s) used for CRL checks in `ssh_certd_config`. If the CA services (OCSP, CRLs) are located behind a firewall, define also the SOCKS server.

```
LdapServers                     ldap://ldap.example.com:389
SocksServer                     socks://fw.example.com:1080
```

4. Restart the server as instructed in Section 3.1.3.

For more information on the configuration file options, see sshd2_config(5) and ssh_certd_config(5).

Certificates Stored in SAF

To configure the server to allow host-based authentication with SAF certificates, perform the following tasks. Replace the names and IDs with those appropriate to your system:

1. In the `/opt/tectia/etc/sshd2_config` file under the **AllowedAuthentications** keyword, add `hostbased` as an allowed method. For example:

```
AllowedAuthentications              hostbased,publickey,password
AuthHostbased.Cert.Required         yes
```

If you want to allow multiple authentication methods, place the least interactive method first. Setting the **AuthHostbased.Cert.Required** option to `yes` defines that the client host must authenticate with a certificate or else the authentication will fail.

2. Define the certificate validation method in the `/opt/tectia/etc/sshd2_config` file. The validation method can be either `saf`, `tectia`, or both (`saf,tectia`).

```
AuthHostbased.Cert.ValidationMethods    tectia|saf|saf,tectia
```

If `saf` is specified, RACF/SAF is used for validating client host certificates. The host certificates must be found in a trusted key ring defined by the **KnownHostsEkProvider** keyword. Note that when only SAF validation is used, the certificate validity period and revocation status are not checked.

If `tectia` is specified (or the keyword is missing from the configuration), the Tectia Certificate Validator (**ssh-certd**) is used for validating client host certificates. The host certificates must be issued by a trusted certification authority defined in the **HostCA**, **HostCAEkProvider**, or **HostCAEkProviderNoCRLs** keyword of `ssh_certd_config`.

If both values are specified, the RACF/SAF validation is performed first and after that the Tectia validation. The host certificates must be found in a trusted key ring defined by the `KnownHostsEkProvider` keyword. Also the CA certificate of the issuing certification authority must be found in a trusted key ring defined by the `HostCAEkProvider` or `HostCAEkProviderNoCRLs` keyword of `ssh_certd_config`.

> **ℹ Note**
>
> Several of the following steps are marked either as *SAF validation* or *Tectia validation*. These steps are alternatives depending on the value of `AuthHostbased.Cert.ValidationMethods`. If `saf,tectia` is selected, both sets of steps need to be completed.

3. (*Tectia validation*) Get the CA certificate and store it to a data set, for example `'HOSTCA.CRT'`.

4. (*Tectia validation*) To add the CA certificate into SAF, give the following TSO commands:

```
RACDCERT CERTAUTH ADD('HOSTCA.CRT')TRUST WITHLABEL('HOSTCA')
RACDCERT ID(SSHD2) ADDRING(SSH-HOSTCA)
RACDCERT ID(SSHD2) CONNECT(CERTAUTH LABEL('HOSTCA') RING(SSH-HOSTCA)
  USAGE(CERTAUTH))
RACDCERT ID(SSHD2) LISTRING(SSH-HOSTCA)
```

5. (*Tectia validation*) For the settings to take effect, give the following TSO command:

```
SETROPTS RACLIST(DIGTCERT) REFRESH
```

6. (*SAF validation*) Get the client host certificate and store it to some data set, for example `'CLIENT1.CRT'`.

7. (*SAF validation*) To add the client host certificate into SAF, give the following TSO commands:

```
RACDCERT ID(SSHD2) ADD('CLIENT1.CRT') TRUST WITHLABEL('CLIENT1')
RACDCERT ID(SSHD2) ADDRING(SSH-KNOWNHOSTS)
RACDCERT ID(SSHD2) CONNECT(ID(SSHD2) LABEL('CLIENT1')
  RING(SSH-KNOWNHOSTS) USAGE(PERSONAL))
RACDCERT ID(SSHD2) LISTRING(SSH-KNOWNHOSTS)
```

8. (*SAF validation*) For the settings to take effect, give the following TSO command:

```
SETROPTS RACLIST(DIGTCERT) REFRESH
```

9. (*SAF validation*) If only SAF validation is used, define the z/OS SAF external key provider that contains the known host certificates with the `KnownHostsEkProvider` keyword in the `/opt/tec-tia/etc/sshd2_config` file:

```
KnownHostsEkProvider            "zos-saf:KEYS(ID(SSHD2)RING(SSH-KNOWNHOSTS))"
```

10. (*Tectia validation*) If Tectia validation is used, define the z/OS SAF external key provider that contains the CA certificates with the `HostCAEkProvider` keyword in the `ssh_certd_config` file:

```
HostCAEkProvider               "zos-saf:KEYS(ID(SSHD2)RING(SSH-HOSTCA))"
```

11. (*Tectia validation*) If Tectia validation is used, define also the LDAP server(s) used for CRL checks in the `ssh_certd_config` file. If the CA services (OCSP, CRLs) are located behind a firewall, define also the SOCKS server.

```
LDAPServers                    ldap://ldap.example.com:389
SocksServer                    socks://fw.example.com:1080
```

Defining the LDAP server is not necessary if the CA certificate contains a `CRL Distribution Point` or an `Authority Info Access` extension.

12. Restart the server as instructed in Section 3.1.3.

For more information on the configuration file options, see sshd2_config(5). For information on the format of the external key initialization string, see ssh-externalkeys(5).

5.8.3 Optional Configuration Settings

To make the host-based authentication more secure, you may want to consider the following optional configuration settings:

- With the **AllowSHosts** and **DenySHosts** keywords in the `sshd2_config` file you can filter the `.shosts`, `.rhosts`, `/etc/hosts.equiv` and `/etc/shosts.equiv` entries.

- If you want to allow only global configuration files (`/etc/hosts.equiv` and `/etc/shosts.equiv`), make sure that you have the following entry in your `sshd2_config` file:

```
IgnoreRhosts                    yes
```

After this modification the `.shosts` and `.rhosts` files will not be used in host-based authentication.

• To force an exact match between the host name that the client sends to the server and the client's DNS entry, make sure that you have the following definition in your `/opt/tectia/etc/sshd2_config` file:

```
HostbasedAuthForceClientHostnameDNSMatch yes
```

In this case, make sure the `/etc/hosts` file has the fully qualified domain name listed before the short host name, for example:

```
123.123.123.123   client.example.com   client
```

Even if you are not using `/etc/hosts` as your primary resolver, you may need to add entries to it for the client and the server to allow them to resolve each other's fully qualified domain names (if they are not able to do so otherwise).

Please note that when **`HostbasedAuthForceClientHostnameDNSMatch`** is used, host-based authentication through NAT (Network Address Translation) will not work.

5.9 User Authentication with Keyboard-Interactive

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

Currently on Tectia Server for IBM z/OS, the supported submethods for keyboard-interactive are `password` and `plugin`.

Methods that require passing some binary information, such as public-key authentication, cannot be used as submethods of keyboard-interactive. But public-key authentication, for example, can be used as an additional method alongside keyboard-interactive authentication.

> **ℹ 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.

Keyboard-interactive is not enabled by default on Tectia Server for IBM z/OS. To set up keyboard-interactive authentication, do the following:

1.   Include the following line in the `/opt/tectia/etc/sshd2_config` file:

```
AllowedAuthentications         keyboard-interactive
```

Also other authentication methods can be listed.

2. The submethods and policy for keyboard-interactive are configured as follows:

```
AuthKbdInt.Required            password
AuthKbdInt.Optional            password,plugin
AuthKbdInt.NumOptional         1
AuthKbdInt.FailureTimeout      2
...
```

The default number of optional submethods that must be passed is 0, although if no required submethods are specified, the client must always pass at least one optional submethod.

See sshd2_config(5) for more information on the keywords.

3. (*Optional*) You can configure password change for the `password` submethod with the two configuration variables described in Section 5.5.

4. Restart the server as instructed in Section 3.1.3.

Chapter 6 System Administration

Secure system administration using shell connection and remote commands is a common use case for Tectia Server for IBM z/OS.

You can limit the services that are accessible via Secure Shell with the server settings. For example, the Secure Shell server may deny terminal access and allow only file transfer or tunneling. The restrictions are described in Section 4.9.1

6.1 Shell Access and Remote Commands

Users can set up their USS shell environments to use a code page other than the default IBM-1047 code page. Tectia Server for IBM z/OS supports conversion of the terminal data stream to several code pages. There are two ways to specify the code page: the **chcp** command and server configuration.

Tectia Server also supports conversion of the line delimiter through server configuration. However, many SSH clients do line delimiter conversion or use LF, which is the Tectia default line delimiter.

The code page of the data on the line can also be specified. It must be the code page that the Secure Shell client uses or is configured to use. ISO8859-1 is the default code page.

6.1.1 Supporting the chcp Command

The **chcp** command can be used to dynamically change the code page in interactive terminal sessions. The default Tectia Server for IBM z/OS configuration supports this usage. The code page (and the locale) is typically set up in each user's `.profile` or `.tcshrc` script (see the IBM book *z/OS USS User's Guide*).

> ### ℹ Note
>
> The `/etc/profile` and the user's `.profile` script must work in both the IBM-1047 code page and the user's code page. Several of the EBCDIC variant characters are significant in shell scripts.

The **chcp** command has code page names as its arguments. The code pages must be supported by **iconv**. The **chcp** command cannot specify a translation table file.

6.1.2 Configuring Terminal Data Conversion

The code pages or translation table for code page conversion and the line delimiters for EOL conversion can be specified in the server configuration files. For interactive sessions, the code pages are the initial values (they can be changed with the **chcp** command).

The configuration variables have defaults which are used if the variables are not specified in any configuration file. The values specified in `sshd2_config` override the defaults and can themselves be overridden by settings in host-specific or user-specific subconfiguration files. For more infomation, see Section 4.2.2 and Section 4.2.1).

Subconfiguration files are read in when a terminal session is started - they can be changed without restarting the server or the client connection.

The related configuration variables are:

ShellAccountCodeset defines the CCS expected by the shell and shell applications.

ShellTransferCodeset defines the coded character set (CCS) on the line.

ShellAccountLineDelimiter defines the line delimiter expected by the shell and shell applications.

ShellTransferLineDelimiter defines the line delimiter on the line.

ShellConvert defines whether conversions are to be done for shell access and remote command execution.

ShellTranslateTable defines the MVS translate table to be used from line to shell.

Chapter 7 File Transfer Using SFTP

This chapter contains instructions on secure file transfer (SFTP) settings with Tectia Server for IBM z/OS. For information on securing the server for file transfer use, see Section 4.9.2.

7.1 Creating a User for Batch File Transfers

You can create users specifically for running Tectia Server for IBM z/OS file transfer batch jobs. Batch users need only an OMVS segment. They do not need TSO or passwords.

To create a user, for example SFTUSER, for running batch file transfers, do the following steps:

1. Create a user, for example using RACF:

```
ADDUSER SFTUSER NAME('SSH Tectia Batch User) OWNER(IBMUSER) +
NOPASSWORD NOOIDCARD +
OMVS(HOME('/u/SFTUSER') PROGRAM( /nologin ) UID(12345))
```

   The home directory and UID must be unique for each user.

2. Enter the following commands:

```
# mkdir /u/SFTUSER              ❶
# mkdir /u/SFTUSER/.ssh2        ❷
# chown -R SFTUSER /u/SFTUSER   ❸
# chmod 700 /u/SFTUSER/.ssh2    ❹
```

   ❶   Create the USS home directory /u/SFTUSER for the SFTUSER user.
   ❷   Under the home directory, create the .ssh2 subdirectory for storing the remote server host keys (and optionally user keys and the user-specific ssh-broker-config.xml configuration file).
   ❸   Make SFTUSER the owner of these directories.
   ❹   Give only SFTUSER full (read, write and execute) permissions to the .ssh2 subdirectory.

> **ⓘ Note**
>
> If you use ACF2 for system security and have enabled the TSO Command Limiting list, the batch user must have `BPXWRTCM` permitted.

7.2 Controlling File Transfer

The current Secure Shell protocol and current clients do 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 coded character set (CCS).

Tectia Server for IBM z/OS needs more information: which transfer format to use, what coded character sets are involved, and what the file characteristics are. When Tectia is used at both ends, the information can be relayed by using the **site** commands of **scpg3** and **sftpg3**, or read from environment variables. When a third-party Secure Shell client is used together with Tectia Server for IBM z/OS, the information can be either encoded in the file name (advice string), read from a file transfer profile, or read from environment variables.

For more information, see *Tectia Server for IBM z/OS User Manual*

7.2.1 Handling Prematurely Ending File Transfers

By default, when a file transfer to Tectia Server for IBM z/OS ends prematurely, the allocated data set is kept.

To change this behavior, you can specify the `--attribute=conddisp` option with **sft-server-g3** in the `sshd2_config` file:

```
subsystem-sftp    /opt/tectia/libexec/sft-server-g3 --attribute=conddisp:DELETE
```

The `conddisp` option can take values `CATLG`, `UNCATLG`, `KEEP`, and `DELETE`, and they have the following effects, depending on the file type (MVS or HFS):

- `CATLG` (default): 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.

7.2.2 Controlling Staging during File Transfers

It is possible to change staging behavior in Tectia server on z/OS. You can specify the `--attribute=staging` option with **sft-server-g3** in the `sshd2_config` file:

```
subsystem-sftp    /opt/tectia/libexec/sft-server-g3 --attribute=staging:YES
```

When `staging` is set to `NO` (default), staging is not used if not explicitly requested by the client.

When `staging` is set to `YES`, staging is used when needed.

Third-party file transfer clients can request staging with the following file transfer advice string:

```
/ftadv:staging=yes/
```

or in short:

```
/ftadv:s=yes/
```

If the file transfer client uses SFTP protocol for transferring the file and accesses the file or data set in correct order, staging is not needed. However, if the client accesses file data in random offsets, staging is necessary when data sets are involved or conversions are used.

7.2.3 Restoring Archived Data Sets

Storage management software, such as data facility hierarchical storage manager (DFHSM), can migrate needless data sets by storing them out of the normal data file system to a tape or some other storage medium. These data sets are listed in the catalog with a special volume serial number, normally `MIGRAT`, but some configurations or products may use different identifiers. A migrated data set is recalled by mounting the offline storage medium on which it is stored and copying it back to an online medium. This process can happen automatically under the storage manager's control, or it may be explicitly requested. There will usually be some delay while recall occurs.

From an SFTP user point of view, it may be desirable to have migrated data sets automatically recalled, or it may be preferred not to do so and ignore them if they would be selected by a globbing pattern. This behavior can be controlled via the `automount` and `autorecall` extended file attributes. See *Tectia Server for IBM z/OS User Manual* for further details.

When allocating a migrated data set, the SFTP subsystem recognizes that the volume serial number `MIGRAT` is purely symbolic and does not use it to specify the volume, allowing the storage manager to do so. In the case where the special volume serial number for a migrated data set is other than `MIGRAT`, it is necessary to inform the SFTP subsystem that this is the case. The environment variable `SSH_SFT_PSEUDOVOLUME_VOLSERS` is provided for this purpose. It may be set to a comma-separated list of one or more pseudo-volume serial numbers, each of which is a string. If the volume serial number of a migrated data set matches one of these, the subsystem will treat the data set exactly as if it had contained `MIGRAT`, that is, it will not specify a volume when attempting to allocate the data set. Note that this is the sole effect of this environment variable: it does not cause migration or recall to happen, or modify the process in any other way. Supply the setting

`SSH_SFT_PSEUDOVOLUME_VOLSERS=<volser_list>` (where `<volser_list>` is a comma-separated list of pseudo-volume serial numbers) as appropriate in any of `/etc/environment`, the user's logon profile or rc script, `~/.ssh2/environment`, or the data set allocated to `STDENV DD` for a batch job.

For recall of migrated data sets to happen when they are requested via SFTP, the following conditions are required:

- The user ID under which SFTP is running must be permitted to recall migrated data sets. If the RACF facility `SSZ.MOUNT` is defined, the user must have read access.

- The extended file attribute `automount` must be set to `yes` or `immed` via **site** command or configuration.

- The `autorecall` extended file attribute must be set or allowed to default to `yes`.

To avoid unintentionally recalling data sets when it is not desired to do so, set the extended file attribute `autorecall` to `no`. If the special volume serial number for migrated data sets is anything other than `MIGRAT`, define the special volume serial number(s) in use via the `SSH_SFT_PSEUDOVOLUME_VOLSERS` environment variable.

Chapter 8 Secure File Transfer Using Transparent FTP Security

In addition to secure file transfer with SFTP command-line clients, Tectia Server for IBM z/OS provides ways to secure existing FTP file transfers transparently. These are transparent FTP tunneling and FTP-SFTP conversion.

Even though settings related to transparent FTP security are made at the FTP client side, they usually require administrator privileges.

8.1 Introduction to Transparent FTP Security

Transparent FTP tunneling and FTP-SFTP conversion can be used to secure both interactive and unattended FTP sessions. They provide a quick and easy way to secure FTP file transfers without the need to change existing FTP jobs or scripts.

Note that it is not possible to do transparent FTP tunnels or FTP-SFTP conversion specifying IPv6 addresses.

8.1.1 Transparent FTP Tunneling

Transparent FTP tunneling is implemented using the Tectia SOCKS Proxy component. Tectia SOCKS Proxy acts as a SOCKS proxy for the FTP client application on the Tectia Server for IBM z/OS host and captures FTP connections based on filter rules. The tunneling is transparent to the user and to the FTP application. The only change needed in the FTP application is to change the SOCKS proxy setting to point to a localhost listener.

The SOCKS Proxy uses the host name, user name, and password information provided by the FTP client application to open an authenticated and encrypted tunnel to a Secure Shell server.

The Secure Shell server can also be defined in the filter rules. In this case, the secure tunnel is terminated at the Secure Shell server and from there the FTP connection is forwarded to the FTP server unsecured.

The principle of transparent FTP tunneling is shown in Figure 8.1. Before starting the tunneling, the Tectia SOCKS Proxy must be running and listening to the SOCKS port 1080 on the File Transfer Client host.
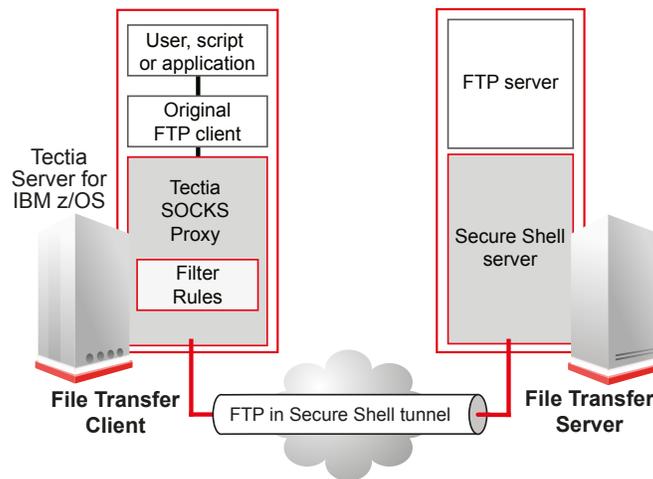

Figure 8.1. Transparent FTP tunneling

The following steps happen in transparent FTP tunneling:

1. An application, a script, or a user triggers a file transfer.

2. The FTP client in the File Transfer Client machine starts a file transfer to the FTP server in File Transfer Server.

3. The FTP client makes a SOCKS query. The SOCKS setting in the FTP client is set to point to the localhost Tectia SOCKS Proxy instead of a real firewall.

4. The filter rules that specify which connections to capture are defined in the SOCKS Proxy configuration. Connections can be captured based on the destination address and/or port.

5. The SOCKS Proxy module creates an authenticated and encrypted Secure Shell tunnel to a Secure Shell server. The user can be authenticated with the FTP user name and password, or with public keys. The Secure Shell server can be the FTP server specified in the original FTP request, or another server can be configured in the filter rules.

6. The secure tunnel is terminated at the Secure Shell server.

7. The Secure Shell server forwards the connection to the FTP Server, and the FTP server can continue with post-processing of the transferred files. If the FTP server is located on a third host, the connection from the Secure Shell server to the FTP server will be unsecured. This is why it is recommended that there is at least one Secure Shell server in each physically secured area, for instance, in a machine room.

8.1.2 FTP-SFTP Conversion

FTP-SFTP conversion is implemented using the Tectia SOCKS Proxy component. Tectia SOCKS Proxy acts as a SOCKS proxy for the FTP client application on the Tectia Server for IBM z/OS host and captures FTP connections based on filter rules. FTP connections are converted to SFTP, transparently to the user and to the FTP application. The only change needed in the FTP application is to change the SOCKS proxy setting to point to a localhost listener.

The SOCKS Proxy uses the host name, user name, and password information provided by the FTP client application to open an authenticated and encrypted SFTP connection to a Secure Shell SFTP server.

The Secure Shell SFTP server can also be defined in the filter rules. This way, the client's request for the FTP server destination can be overridden.

The principle of FTP-SFTP conversion is shown in Figure 8.2. Before starting the conversion, the Tectia SOCKS Proxy must be running and listening on the SOCKS port 1080 on the File Transfer Client host.
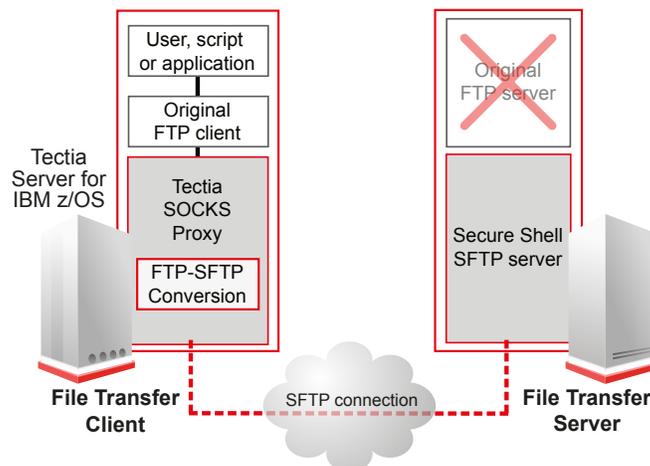
Figure 8.2. The architecture of FTP-SFTP conversion

The following steps happen during the FTP-SFTP conversion:

1. An application, a script, or a user triggers a file transfer.

2. The original FTP client in the File Transfer Client host starts opening a file transfer connection to the original destination FTP server (in File Transfer Server).

3. The FTP client makes a SOCKS query. The SOCKS setting in the FTP client is set to point to the localhost Tectia SOCKS Proxy instead of a real firewall.

4. The filter rules that specify which connections to capture are defined in the SOCKS Proxy configuration. Connections can be captured based on the destination address and/or port.

5. The FTP-SFTP conversion module can extract the user name, password, and the destination host name from the secured FTP application, and use them for authentication and connection setup with the Secure Shell SFTP server.

6. The FTP-SFTP conversion module manages the FTP connection so that it remains unchanged from the original FTP client's point of view. FTP is converted to secure SFTP file transfer.

7. The SFTP connection is managed by the Connection Broker module.

8. The Secure Shell SFTP server in the File Transfer Server host is the end point of the file transfer.

   The unsecured original FTP server program can be eliminated from the server host.

8.1.3 Summary of Configuration Steps

To enable transparent FTP tunneling and FTP-SFTP conversion, you need to complete the following tasks:

1. Configure the Tectia SOCKS Proxy to listen on port 1080 on the client host and define the filter rules in the `ssh-socks-proxy-config.xml` configuration file. See Section 8.2.

2. Create a user, for example, `SSHSP`, for running the SOCKS Proxy. See Section 8.3.

3. Start the `ssh-socks-proxy` process. See Section 8.4.

4. Configure the SOCKS settings for the FTP client. See Section 8.5.

After these tasks are completed, the specified FTP connections will be automatically tunneled or converted to SFTP, transparently to the user.

8.2 Configuring the SOCKS Proxy

The Tectia SOCKS Proxy component used in transparent FTP tunneling and FTP-SFTP conversion does not use the regular client configuration `ssh-broker-config.xml`. Instead, the configuration is read from the `ssh-socks-proxy-config.xml` file.

Like the `ssh-broker-config.xml` file, the `ssh-socks-proxy-config.xml` file can be located in the system-wide `/opt/tectia/etc` directory and/or in the user-specific `$HOME/.ssh2` directory.

In ISPF you can access the SOCKS Proxy configuration file via the Tectia SSH Assistant's option **3.4 SOXP**.

The SOCKS Proxy reads the server host keys from the normal locations (`$HOME/.ssh2` and `$HOME/.ssh2/hostkeys`). However, if the server authenticates itself using a certificate, the information on the CA certificate (or the server certificate) must be defined separately in the `ssh-socks-proxy-config.xml` file.

8.2.1 The `ssh-socks-proxy-config.xml` configuration file

The `ssh-socks-proxy-config.xml` configuration file uses the same format as the `ssh-broker-config.xml` configuration file.

For a detailed description of the elements used in the configuration, see Appendix *Connection Broker and SOCKS Proxy Configuration Files* in *Tectia Server 6.6 for IBM z/OS User Manual*.

An example configuration file shown below tunnels all FTP connections using the host name and user name provided by the FTP client application. To enable the configuration, copy the text to `/opt/tectia/etc/ssh-socks-proxy-config.xml`. You can edit the `ssh-socks-proxy-config.xml` configuration file using a normal text editor.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE secsh-broker SYSTEM
 "/opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd">
<secsh-broker version="6.6" >
  <default-settings>
    <authentication-methods>
      <auth-password />
      <auth-keyboard-interactive />
    </authentication-methods>
  </default-settings>
  <profiles>
    <profile name="dynamic-ftp"
             id="id1"
             host=""
             port="22"
             user="">
    </profile>
  </profiles>
  <static-tunnels>
    <tunnel type="socks-proxy"
            listen-address="127.0.0.1"
            listen-port="1080"
            dst-port="0"
            profile="" />
  </static-tunnels>
  <filter-engine>
    <rule ip-address=".*"
          ports="21"
          action="ftp-tunnel"
          profile-id="id1"
          username-from-app="YES"
          hostname-from-app="YES"
          fallback-to-plain="NO" />
  </filter-engine>
  <logging>
    <log-events facility="auth" severity="informational">
      Connector_filter_rule
```

```
    </log-events>
  </logging>
</secsh-broker>
```

The following settings are required in the `ssh-socks-proxy-config.xml` file to enable transparent FTP tunneling or FTP-SFTP conversion:

default-settings

The `default-settings` element defines, for example, the user authentication methods used by the Tectia SOCKS Proxy. Requiring password authentication ensures that the password information provided by the FTP client application is used.

profiles

At least one `profile` element must be defined.

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 must not contain any blanks/spaces.

The `host` attribute defines the address of the Secure Shell server host. If it is left empty and/or under the `filter-engine/rule` element `hostname-from-app="yes"`, the Secure Shell connection is opened to the destination host given in the SOCKS request. Otherwise the Secure Shell connection is opened to the host specified in the profile and in FTP tunneling, FTP connections are forwarded to the requested hosts.

The `port` attribute specifies the Secure Shell server port. The default port is `22`.

The `user` attribute specifies the user name on the Secure Shell server. If it is left empty, the user name given by the FTP client is used when opening the Secure Shell connection.

static-tunnels

At least one `tunnel` element must be defined.

For transparent tunneling, the tunnel `type` must be set as `"socks-proxy` and the `port` as `"1080"`.

The `listen-address` is usually the loopback address `"127.0.0.1"`, but can be an address of any local interface that will be listened.

The `dst-port` attribute is set to `0` and the `profile` attribute is left empty when transparent tunneling and FTP-SFTP conversion are used.

filter-engine

At least one `rule` element must be defined.

The `ip-address` attribute specifies the target host IP address to be filtered. It can be a regular expression. Connections to the specified address are captured. With transparent FTP tunneling and FTP-SFTP conversion, this can be usually set to capture all connections (`".*"`), as the connections are already filtered by the SOCKS Proxy settings.

The `ports` attribute specifies the ports to be filtered. It can be a single port or a range. A range is specified with a dash between two integers (such as `"21-25"`).

The `action` attribute specifies the action to be done when a filter is used. For transparent FTP tunneling, the action is `"ftp-tunnel"`. For FTP-SFTP conversion, the action is `"ftp-proxy"`.

The `profile-id` attribute is a reference to a `profile` element and should contain the same value as the `id` attribute of the profile.

The `hostname-from-app` attribute defines whether the SOCKS Proxy 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`. With Tectia SOCKS Proxy on z/OS, this is usually set to `"yes"`. 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. With Tectia SOCKS Proxy on z/OS, this is usually set to `"yes"`. This setting will override any user name settings made in a related connection profile.

When applying the filter rule, if creating the tunnel fails or the connection to the Secure Shell server fails, the SOCKS Proxy will normally return a "host not reachable" error. However, if the `fallback-to-plain` attribute is set to `"yes"`, a direct (unsecured) connection is used instead.

8.2.2 Storing Remote Server Host Keys

When opening the transparent tunnel or an SFTP session with FTP-SFTP conversion, accepting new or changed server host keys cannot be prompted from the user. In addition, transparent FTP tunneling and FTP-SFTP conversion always use the IP address of the Secure Shell server when opening the secure tunnel. This means that the host keys of the Secure Shell tunneling servers must be stored beforehand based on the IP addresses of the servers.

The keys can be stored by connecting to each host individually with the IP address of the host using an interactive shell and accepting the host keys one by one, or by using the **ssh-keydist-g3** key distribution tool. More information and examples on storing remote server keys can be found in *Tectia Server 6.6 for IBM z/OS User Manual*.

Disabling Host Key Check

As an alternative to storing the remote server host keys, it is possible to disable the host key checking entirely. To do this, set the `auth-server-publickey` element's `policy` attribute to `"advisory"` in the `ssh-socks-proxy-config.xml` file.

⚠ **Caution**

> Consider carefully before setting the policy to `advisory`. Disabling the host key checks makes you vulnerable to man-in-the-middle attacks.

In the following example the `auth-server-publickey` element is defined under default settings (`default-settings/server-authentication-methods`). It can also be defined per connection profile (under `profiles/profile/server-authentication-methods`).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE secsh-broker SYSTEM
 "/opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd">
<secsh-broker version="6.6" >

  <default-settings>
    <server-authentication-methods>
      <auth-server-publickey policy="advisory" />
    </server-authentication-methods>
  </default-settings>

  <profiles>
...
```

For more information on the host key policy settings, see Appendix *Connection Broker and SOCKS Proxy Configuration Files* in *Tectia Server 6.6 for IBM z/OS User Manual*.

8.3 Creating the `SSHSP` User

If `ssh-socks-proxy` is going to be run as a started task, you need to create a user for running it.

The SSHSP user can be created by running the job generated by Tectia SSH Assistant action **1.4 ADDSOXPU**.

8.4 Running the SOCKS Proxy

The SOCKS Proxy component consists of two processes:

ssh-socks-proxy

The Tectia SOCKS Proxy process that needs to be running before transparent tunneling connections can be made. The Tectia SOCKS Proxy started task can be controlled with the Tectia SSH Assistant ISPF application and with console modify commands. The process can also be started under USS or by using a JCL script.

For more information on the command-line options of **ssh-socks-proxy**, see the description of **ssh-broker-g3** in *Tectia Server 6.6 for IBM z/OS User Manual* Appendix *Command-Line Tools and Man Pages*. (Running **ssh-socks-proxy** will actually run **ssh-broker-g3** in the SOCKS Proxy mode, using the ssh-socks-proxy-config.xml configuration files and with connection caching disabled.)

ssh-socks-proxy-ctl

Control process for the SOCKS Proxy. It can be used, for example, to view the status of the SOCKS Proxy, to reconfigure or stop the SOCKS Proxy, or to load private keys to memory.

For more information on the **ssh-socks-proxy-ctl** options and commands, see the description of **ssh-broker-ctl** in *Tectia Server 6.6 for IBM z/OS User Manual* Appendix *Command-Line Tools and Man Pages*.

In the Tectia SSH Assistant ISPF application you can control the SOCKS Proxy (**ssh-socks-proxy**) via submenu **4.3 TSXP**.

Figure 8.3. Tectia SSH Assistant ISPF application - Tasks: Socks Proxy Server (4.3 TSXP)

Table 8.1 contains a summary of the modify commands and Tectia SSH Assistant options for controlling the SOCKS Proxy. They are described in more detail in the following sections.

Table 8.1. Controlling the SOCKS Proxy server

Command	Console	ISPF
Start	`S SSHSP`	`4.3.1 TSXPS`
Stop	`F SSHSP,STOP` or `P SSHSP`	`4.3.2 TSXPP`
Query version	`F SSHSP,VERSION`	`4.3.5 TSXPQV`
Set trace level	`F SSHSP,DEBUG <debug-level>`	`4.3.6 TSXPTR`
Set options for starting	`S SSHSP,OPTS='<options>'`	`4.3.7 TSXPOP`
Get status information	`F SSHSP,STATUS/ST`	`4.3.9 TSXPST`
Get cryptographic algorithms information	`F SSHSP,STATUS/ST ALGORITHMS/ALGS`	`4.3.10 TSXPAL`
List open channels	`F SSHSP,LIST-CHANNELS/LIST-CH/LCH`	`4.3.11 TSXPLCH`
List open connections	`F SSHSP,LIST-CONNECTIONS/LIST-CO/LC`	`4.3.12 TSXPLC`
Get the status of a connection	`F SSHSP,CONNECTION-STATUS/CONNECTI/CS`	`4.3.13 TSXPCS`
Close a connection	`F SSHSP,CLOSE-CONNECTION/CLOSE-CO/CC`	`4.3.14 TSXPCC`

The console messages generated by the SOCKS Proxy are listed in Table E.2.

8.4.1 Starting the SOCKS Proxy

To run Tectia SOCKS Proxy as a started task, you can use the JCL procedure `USER.PROCLIB(SSHSP)` (note that the name, defined in 0.2 SETO, might differ from this default name). Before running the job, a user must be created for running the task (as described in Section 8.3). The user can be created by running the job generated by Tectia SSH Assistant action **1.4 ADDSOXPU**.

Console

Start the SOCKS Proxy with the following operator command:

```
===> S SSHSP
```

The `ssh-socks-proxy` job starts.

ISPF

To start the SOCKS Proxy, enter option **4.3.1 TSXPS (Start the SOCKS Proxy server)**.

You should see the following console message:

```
ISF031I CONSOLE <USERID> ACTIVATED
-S SSHSP
 +SSZ3006I Task ssh-socks-proxy started
```

USS

Under USS, Tectia SOCKS Proxy can be started by running the following command:

```
> /opt/tectia/bin/ssh-socks-proxy
```

For more information on the command-line options of **ssh-socks-proxy**, see the description of **ssh-broker-g3** in Appendix *Command-Line Tools and Man Pages* of *Tectia Server 6.6 for IBM z/OS User Manual*.

8.4.2 Stopping the SOCKS Proxy

Console

To stop the SOCKS Proxy under MVS when you are running it as a started task, enter:

```
===> F SSHSP,STOP
```

OR:

```
===> P SSHSP
```

ISPF

To stop the SOCKS Proxy in ISPF, enter the Tectia SSH Assistant option **4.3.2 TSXPP (Stop the SOCKS Proxy server)**.

You should see the following console message:

```
ISF031I CONSOLE <USERID> ACTIVATED
-P SSHSP
 +SSZ3003I Command STOP accepted
```

USS

To stop the SOCKS Proxy under USS, execute one of the following commands:

```
> /opt/tectia/bin/ssh-socks-proxy --exit
```

OR:

```
> /opt/tectia/bin/ssh-socks-proxy-ctl stop
```

8.4.3 Querying the SOCKS Proxy Version

Console

You can query the version of the SOCKS Proxy with the following console command:

```
===> F SSHSP,VERSION
```

ISPF

To check the version of the SOCKS Proxy in ISPF, enter the Tectia SSH Assistant option **4.3.5 TSXPQV (Query the version of the running SOCKS Proxy server)**.

8.4.4 Reloading ssh-socks-proxy configuration

If you make changes to the `ssh-socks-proxy-config.xml` configuration file the changes will not take effect until the SOCKS Proxy configuration is reloaded. After you have reloaded the SOCKS Proxy configuration, existing connections will continue to use the old configuration settings while new connections will use the reconfigured settings.

You can reload the SOCKS Proxy to the current configuration file by executing the following command:

```
> /opt/tectia/bin/ssh-socks-proxy-ctl reload
```

8.4.5 Setting Options for Starting the SOCKS Proxy

Console

As an OPTS parameter, you can give parameters that the actual SOCKS Proxy binary accepts.

For more information on the command-line options of **ssh-socks-proxy**, see the description of **ssh-broker-g3** in *Tectia Server 6.6 for IBM z/OS User Manual* Appendix *Command-Line Tools and Man Pages*. (Running **ssh-socks-proxy** will actually run **ssh-broker-g3** in the SOCKS Proxy mode, using the `ssh-socks-proxy-config.xml` configuration files and with connection caching disabled.)

For example:

```
===> S SSHSP,OPTS='-D 9'
```

ISPF

To set options for starting the SOCKS Proxy in ISPF, enter the Tectia SSH Assistant option **4.3.7 TSXPOP**.

The available options are described in detail in the description of **ssh-broker-g3** in *Tectia Server 6.6 for IBM z/OS User Manual* Appendix *Command-Line Tools and Man Pages*.

8.4.6 More Modify Commands for the SOCKS Proxy Server

The following modify commands are also available for the running SOCKS Proxy:

CLOSE-CONNECTION/CLOSE-CO/CC
    Closes the defined connection. You can also enter multiple connection IDs to close several connections.

CONNECTION-STATUS/CONNECTI/CS
    Displays a detailed connection status for the connection ID (the numeric identifier shown by the LIST-CONNECTIONS) command.

LIST-CHANNELS/LIST-CH/LCH

> 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-CONNECTIONS/LIST-CO/LC

> Displays a list of the currently open connections together with connection parameters and traffic statistics. Displays also the connection ID which can be used with other commands to identify the connection.

STATUS/ST [ALGORITHMS/ALGS]

> Without parameters, displays short statistics and a configuration summary for the currently running SOCKS Proxy.
>
> Options:
>
> ALGORITHMS/ALGS
>
> > Displays the supported cryptographic algorithms for key exchange, host key, ciphers, MACs, compression, and user authentication.

The same commands can be executed using the options in the Tectia SSH Assistant submenu **4.3.8 TSXPMO**.

Figure 8.4. Tectia SSH Assistant ISPF application - More modify commands for the SOCKS Proxy server (4.3.8 TSXPMO)

8.5 Configuring FTP

To enable transparent FTP tunneling, the SOCKS setting in the FTP application needs to be pointed to a localhost listener.

If you are running a real SOCKS proxy on your z/OS system and the localhost port 1080 is already reserved, you can add a second loopback address to your TCPIP profile for the transparent FTP tunneling proxy.

To add the secondary IP address, edit the TCPIP profile on your z/OS system. Under the HOME heading, add the following line:

```
127.0.0.2      LOOPBACK
```

Make it the second IP address, or put it lower down, because the first address is the machine's default IP address. Follow the procedure shown in ZOS TCPIP Setup. Check that the address is there with the following TSO command:

```
===> NETSTAT HOME
```

8.5.1 Editing the FTP Client Configuration

To make the FTP client use SOCKS, add or edit the FTP client configuration data set (or file). The FTP client configuration can be defined in several places (globally or per user) and the first configuration found is used. The search order for the configuration is listed in the IBM document *z/OS Communication Server: IP Configuration Reference*. Make sure that you define the configuration in a location that has the highest precedence on your system.

In the FTP client configuration data set (for example, `userid`.`FTP`.`DATA`), add the following line that specifies the SOCKS configuration file to be used, for example:

```
SOCKSCONFIGFILE /etc/socks.conf
```

The example above specifies a SOCKS configuration that is usable in the whole system. The SOCKS configuration can also be user-specific, for example `$HOME/socks.conf`.

The SOCKS configuration can also be a data set, for example:

```
SOCKSCONFIGFILE TSTUSR.SSHFTP.SOCKS.CONF
```

FTP clients must use PASSIVE or EXTENDED PASSIVE mode for file transfers. ACTIVE mode is not supported with transparent FTP tunneling. To activate PASSIVE mode, add the following line to the FTP client configuration data set:

```
FWFRIENDLY     TRUE
```

> **ⓘ Note**
>
> The FTP client configuration setting `PASSIVEIGNOREADDR` is set to `FALSE` by default, enabling FTP PASSIVE mode. This setting is required for the SOCKS Proxy to work and it should not be changed in the FTP client configuration data set if the SOCKS Proxy is to be used.

8.5.2 Creating the SOCKS Configuration

Create or edit the SOCKS configuration file or data set you have defined in the FTP configuration (for example, `/etc/socks.conf`).

The following configuration secures connections to hosts 192.168.10.5 and 192.168.10.10, and to class C network 192.168.20.0. Connections to other addresses will be direct:

```
sockd @=127.0.0.1 192.168.10.5 255.255.255.255
sockd @=127.0.0.1 192.168.10.10 255.255.255.255
sockd @=127.0.0.1 192.168.20.0 255.255.255.0
direct 0.0.0.0 0.0.0.0
```

The following configuration assumes that the loopback address for transparent FTP tunneling proxy was changed. It makes all FTP connections use SOCKS on the new loopback address:

```
sockd @=127.0.0.2 0.0.0.0 0.0.0.0
```

8.6 Examples of Transparent FTP Security

This section gives an example of system-wide settings for transparent FTP tunneling and an example of FTP-job-specific settings for transparent FTP tunneling and FTP-SFTP conversion.

The steps assume that the SSHSP user created in Section 8.3 is used for running the Tectia SOCKS Proxy.

Before attempting transparent tunneling, test that the connection to the tunneling server works using **sshg3**.

8.6.1 System-Wide Transparent FTP Tunneling with Fallback

In this example, transparent FTP tunneling is configured to secure file transfers to three different subnets (`10.70.20.0/24`, `172.16.8.0/21`, and `192.168.93.0/25`). Configuration is done system-wide affecting all FTP batch jobs and interactive sessions to those three subnets. FTP connections to other addresses are not affected.

In this example, Tectia SOCKS Proxy is configured to fall back to plain FTP if securing the connection fails. Fallback is a usable feature during the migration phase but must be turned off once all connections are working correctly.

Do the following steps:

1. Copy the `/opt/tectia/etc/ssh-socks-proxy-config-example.xml` configuration file to `/opt/tectia/etc/ssh-socks-proxy-config.xml` (if it does not exist yet).

2. Edit the file to enable fallback to plaintext FTP for transparent FTP tunneling:

```
<filter-engine>
  <rule ip-address=".*"
        ports="21"
        action="ftp-tunnel"
        profile-id="id1"
        username-from-app="YES"
        hostname-from-app="YES"
        fallback-to-plain="YES" />
</filter-engine>
```

3.   (Re)start the SOCKS Proxy.

4.   Edit the system-wide FTP configuration file `SYS1.TCPPARMS(FTPDATA)` to include the following lines:

```
SOCKSCONFIGFILE   EXAMPLE.SYSTEM.SOCKS.CONF
FWFRIENDLY        TRUE
```

5.   Create a new SOCKS configuration data set `EXAMPLE.SYSTEM.SOCKS.CONF` with the following contents:

```
sockd @=127.0.0.1 10.70.20.0 255.255.255.0
sockd @=127.0.0.1 172.16.8.0 255.255.248.0
sockd @=127.0.0.1 192.168.93.0 255.255.255.128
direct 0.0.0.0 0.0.0.0
```

8.6.2 JCL-Specific Transparent FTP Tunneling or FTP-SFTP Conversion

In this example, transparent FTP tunneling or FTP-SFTP conversion is configured per FTP JCL job using a SYSFTPD DD statement.

Do the following steps:

1.   Copy   the   `/opt/tectia/etc/ssh-socks-proxy-config-example.xml`   configuration   file   to
     `/opt/tectia/etc/ssh-socks-proxy-config.xml`.

2.   (*FTP-SFTP conversion only*) If you want to use FTP-SFTP conversion, edit the configuration file and
     change the value of the `action` attribute to `"ftp-proxy"`:

```
  <filter-engine>
    <rule ip-address=".*"
          ports="21"
          action="ftp-proxy"
          profile-id="id1"
          username-from-app="YES"
          hostname-from-app="YES"
          fallback-to-plain="NO" />
  </filter-engine>
```

3.   Start the SOCKS Proxy if it is not already running.

4.   Create a new FTP configuration file `EXAMPLE.SSHFTP.FTPDATA` with the following contents:

```
SOCKSCONFIGFILE   EXAMPLE.JCL.SOCKS.CONF
FWFRIENDLY        TRUE
```

5.   Create a new socks configuration data set `EXAMPLE.JCL.SOCKS.CONF` with the following contents:

```
sockd @=127.0.0.1 0.0.0.0 0.0.0.0
```

6. Modify your existing FTP JCL to use SOCKS by adding a SYSFTPD DD statement, for example:

```
//FTP       EXEC PGM=FTP,PARM='company.example.com
(EXIT=8'
//SYSPRINT  DD  SYSOUT=*
//SYSFTPD   DD  DSN=EXAMPLE.SSHFTP.FTPDATA,DISP=SHR
//SYSIN     DD  *
userid   passwd
ascii
get test.file 'USERID.FTP.TEST'
quit
/*
```

Chapter 9 Tunneling

Tunneling is a way to forward otherwise unsecured TCP 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.

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 (incoming) tunnels, the ports under 1024 (the well-known service ports) are not allowed for the regular users, but are available only for system administrators (root privileges).

There are two basic kinds of tunnels: local (outgoing) and remote (incoming). Agent forwarding is a special case of a remote tunnel.

9.1 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 the following:

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

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

Figure 9.1 shows the different hosts and ports involved in local tunneling.
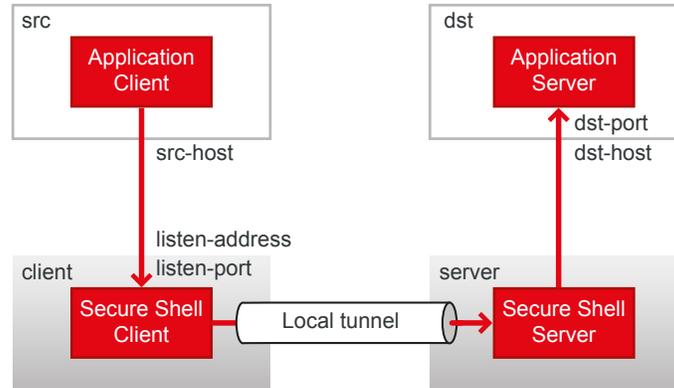
Figure 9.1. Local tunneling terminology

For example, when you issue the following command, all traffic coming to port 1234 on the client will be forwarded to port 23 on the server. See Figure 9.2.

```
$ sshg3 -L 1234:localhost:23 username@sshserver
```
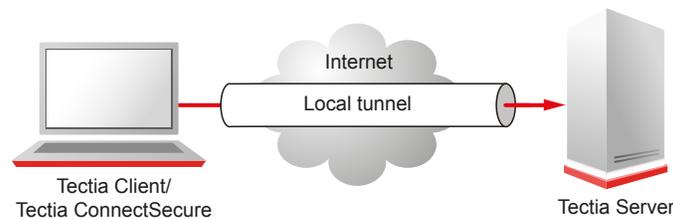

Figure 9.2. Simple local (outgoing) tunnel

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`).

If you have three hosts, for example, `sshclient`, `sshserver`, and `imapserver`, and you forward the traffic coming to the `sshclient` port 143 to the `imapserver` port 143, only the connection between `sshclient` and `sshserver` will be secured. The command you use would be similar to the following:

```
$ sshg3 -L 143:imapserver:143 username@sshserver
```

Figure 9.3 shows an example where the Secure Shell server resides in the DMZ network. The connection is encrypted from the Secure Shell client to the Secure Shell server and continues unencrypted in the corporate network to the IMAP server.
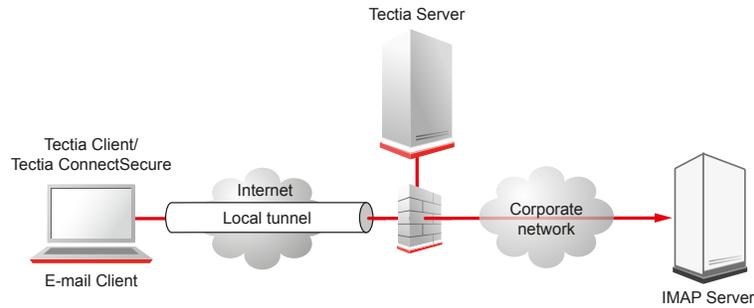
Figure 9.3. Local (outgoing) tunnel to an IMAP server

By default, the server allows local tunnels to all addresses for all users. To restrict tunneling for all or for specified users, see Section 4.9.3.

9.1.1 Tunneling TN3270

When Tectia ConnectSecure on Windows workstations is used together with Tectia Server for IBM z/OS on mainframe, TN3270 application connections can be tunneled transparently.

The administrator specifies tunneling rules for the TN3270 application connection(s) that need to be secured. Alternatively, it is possible to require that all terminal connections initiated by a certain terminal emulator will be tunneled.

When the terminal client accesses a remote mainframe, Tectia ConnectSecure captures the connection transparently and establishes a secure tunnel between the workstation and IBM z/OS system. All TN3270 application connection traffic is then transmitted over an encrypted Secure Shell tunnel, ensuring confidentiality of passwords and application data.

9.2 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 the following:

```
$ sshg3 -R [protocol/][listen-address:]listen-port:dst-host:dst-port server
```

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

Figure 9.4 shows the different hosts and ports involved in remote tunneling.
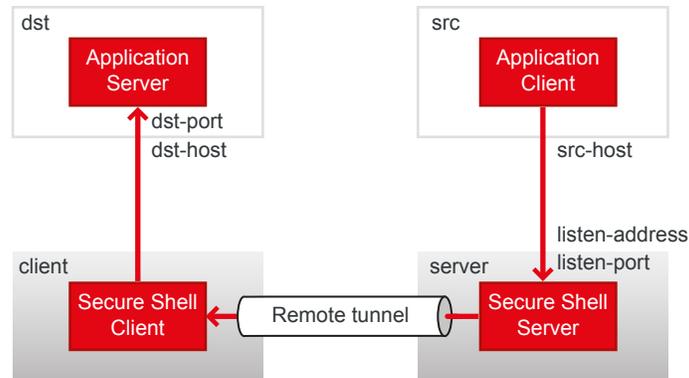
Figure 9.4. Remote tunneling terminology

For example, if you issue the following command, all traffic coming to port 1234 on the server will be forwarded to port 23 on the client. See Figure 9.5.

```
$ 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.
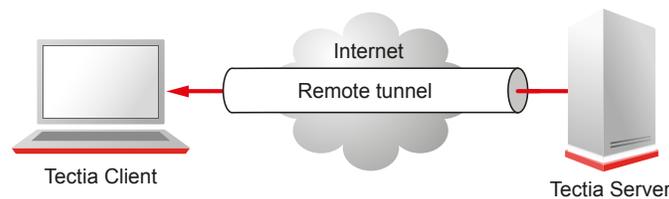
Figure 9.5. Remote (incoming) tunnel

By default, the server allows remote tunnels from all addresses for all users. To restrict tunneling for all or for specified users, see Section 4.9.3.

9.3 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.

By default, Tectia Server for IBM z/OS allows agent forwarding. To deny agent forwarding, set the **AllowAgentForwarding** or **ForwardAgent** keyword in the sshd2_config configuration file to `no`.

Chapter 10 Troubleshooting Tectia Server for IBM z/OS

This chapter gives instructions on debugging and lists some common error situations that may arise with Tectia Server for IBM z/OS.

You can use a troubleshooting tool **ssh-troubleshoot** to collect system information, such as platform version, patches, configuration settings, installed software components, and the current environment and state; and information 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 file that you can send to SSH technical support for analysis to help in troubleshooting situations. See description of the **ssh-troubleshoot** tool in *Tectia Server 6.6 for IBM z/OS User Manual*.

See the Release Notes for known issues in this release of Tectia Server for IBM z/OS.

10.1 Debugging Tectia Server for IBM z/OS

The server can be debugged by using either the `-d` or `-D` option.

The `-d` option launches the server to one-shot mode. In this mode, the server accepts only one connection and exits after the session is disconnected. The server does not spawn a separate task for the connection.

The `-D` option launches the server to continuous debug mode. In this mode, the server keeps on listening to the TCP port and accepts several connections. The server spawns a new task for each new connection and needs to be stopped manually when you want to finish the debugging.

The debug level is either a number, or a comma-separated list of assignments of the format `ModulePattern=de-bug_level`, for example `"*=7,sshd2=2"`. Debug level 4 is usually sufficient. For more detailed information, debug level 7 can be used.

10.1.1 Setting the Debug Level

Setting the Debug Level in the Console

When **sshd2** is run as a started task, the debug level of the running server can be set using the z/OS console modify command **debug**.

To set the debug level of the running server, issue the following command (where *debuglevel* is a number from 1 to 99 indicating the level of diagnostics information required) from the z/OS console:

```
===> F SSHD2,DEBUG debuglevel
```

For example, to set debug level 4 (which is in most cases sufficient):

```
===> F SSHD2,DEBUG 4
```

To disable server debug, set debug level to 0 as follows:

```
===> F SSHD2,DEBUG 0
```

Setting the Debug Level in ISPF

To set the debug level of the server in ISPF, enter the Tectia SSH Assistant option **4.1.6 TSRVTR (Turn trace on or off in the running SSH server)**.

The trace level is a number which directs the server to produce diagnostic trace information on restart. Enter a trace level number in the Options field provided.

Trace level 2 provides user-oriented messages.

To disable server debug, set trace level number to 0.

Figure 10.1. Tectia SSH Assistant ISPF application - Turn trace on or off in the running SSH server (4.1.6 TSRVTR)

10.1.2 Debugging Using USS shell

When debugging the server, a second server daemon can be started to a different TCP port or alternatively the existing server can be stopped and the same port can be used for debugging.

Debugging can be started with the following command (run as the SSHD2 user):

```
> /opt/tectia/sbin/sshd2 -d 4 -p 1234
```

In the command, -d defines the debug level and -p the TCP port. With this command the debug output is displayed to the screen.

Debug can also be forwarded to a file:

```
> /opt/tectia/sbin/sshd2 -d 4 -p 1234
> /tmp/sshd2_debug.out 2>&1
```

Debugging for the SFTP server subsystem can be controlled with the SSH_SFTP_DEBUG and SSH_SFTP_DE-BUG_FILE environment variables. SSH_SFTP_DEBUG defines the debug level for the file transfer server. SSH_SFTP_DEBUG_FILE defines the output file for the debug messages. See Section 10.1.3 below.

10.1.3 Debugging File Transfer

Tectia Server for IBM z/OS uses a separate process, **sft-server-g3**, for file transfer operations.

To obtain debug information from **sft-server-g3**, you can use the environment variables SSH_SFTP_DEBUG, SSH_SFTP_DEBUG_FILE, and SSH_DEBUG_FMT.

When the debugging is enabled, the **sft-server-g3** debug messages are by default sent to the standard error that goes also to the SFTP client. If you want to forward the **sft-server-g3** debug messages into a file and not to the client, you can add the following two environment variables affecting the secure file transfer user into the /etc/environment file or the user-specific $HOME/.ssh2/environment file on the server:

- SSH_SFTP_DEBUG defines the debug level that controls the messages that the **sft-server-g3** process will be showing while executing. To get some debug data, you can use, for example:

  ```
  SSH_SFTP_DEBUG=Sftp=12
  ```

  To get more details, you can use, for example:

  ```
  SSH_SFTP_DEBUG=Sftp*=12,SshFile*=12
  ```

  To get even deeper level of detail, you can use, for example:

  ```
  SSH_SFTP_DEBUG=Sftp*=15,Ssh*File*=15,Ssh*Dir*=15
  ```

  To get full debugging information from the SFTP server, use:

  ```
  SSH_SFTP_DEBUG=Sftp*=20,Ssh*File*=20,Ssh*Dir*=20
  ```

To get most low-level z/OS debugging information:

```
SSH_SFTP_DEBUG=*Mvs*=17
```

To get all low-level z/OS debugging information:

```
SSH_SFTP_DEBUG=*Mvs*=99
```

To get all debugging information related to data set allocation:

```
SSH_SFTP_DEBUG=SshFileMvsAllocate=99
```

To get all debugging information related to z/OS catalog:

```
SSH_SFTP_DEBUG=SshFileMvsCatalog=99
```

- `SSH_SFTP_DEBUG_FILE` defines the file where the debug messages from the **sft-server-g3** will be printed. The value can be, for example:

```
SSH_SFTP_DEBUG_FILE=/tmp/sft_debug.txt
```

  Note that the path must be given without quotation marks. If this variable is not defined, the messages will be sent to standard error and the SFTP client will receive them.

- `SSH_DEBUG_FMT` defines the format of the debug messages. The default variable used is:

```
SSH_DEBUG_FMT=%Dd/%Dt/%Dy %Dh:%Dm:%Ds:%Df %m/%s:%n:%f %M
```

  For more information on the syntax of this variable, see the description of **sftpg3** in Appendix *Command-Line Tools and Man Pages* in *Tectia Server 6.6 for IBM z/OS User Manual*.

On the client side, it is also possible to use the **debug** command of **sftpg3** to get debugging information. For example, to get basic debugging data, give the command:

```
sftp> debug Sftp*=2
```

The same debug strings as with `SSH_SFTP_DEBUG` can also be used.

10.2 Solving Problem Situations

The following sections give workaround instructions for a few problem situations that may occur with Tectia Server for IBM z/OS.

10.2.1 Using the OMVS Shell

INPUT

Appears as a status indicator.

When invoking long-running commands in the OMVS shell, and when the status indicator changes to INPUT, press PF10 to keep the command running.

CEE3536S Not enough storage was available for the WSA

To enable the user to open the shell or other daemon, you need to increase the maximum region size (in bytes) for the address space of the WSA process. You can set a system-wide limit in BPXPRMxx and then set higher limits for individual processes.

For OMVS, increase the size of the address space to 75MB in the RACF user profile. Use the RACF ADDUSER or ALTUSER command to specify the ASSIZEMAX limit on a per-process basis as follows:

```
ALTUSER userid OMVS(ASSIZEMAX(75000000)
```

For more detailed instructions, see Section 3.4.

10.2.2 Common MVS Error Messages

BPXP015I ... PROGRAM ... IS NOT MARKED PROGRAM CONTROLLED

The program is not marked program-controlled. Depending on the program, the following steps may help:

• If the program is **sshd2**, run the following:

```
>extattr +p /opt/tectia/sbin/sshd2
```

• If the program is CEE.SCEERUN2, see Section 2.1.2.

• If the program is **sshg3**, the reason may be that the **sshg3** client does not work if su is used before establishing the Secure Shell connection. The error output depends on the terminal used and what kind of an su is made before the **sshg3** connection attempt. **sshg3** does not need to be program-controlled, instead avoid using su before starting **sshg3**.

10.2.3 Common Tectia Server for IBM z/OS Error Messages

Environment variable '_BPXK_AUTOCVT' is not set, **Environment variable '_BPXK_AUTOCVT' is incorrect**

This environment variable must always be set to a specific value when running Tectia Server for IBM z/OS programs (see Section 3.3). When running under BPXBATCH use the SSHENV (located in *<HLQ>*.V6611.PARMLIB(SSHENV)) data set on a STDENV DD statement . Before running client programs from a USS command prompt, source the sshsetenv script (located in /opt/tectia/doc/zOS/samples).

sshd2[50397305]: FATAL ERROR: setuid: EDC5157I An internal error has occurred., **sshd2[50397305]: FATAL: setuid: EDC5157I An internal error has occurred.**

If a login attempt ends with this error message from the server, the reason may be that **sshd2** is not marked program-controlled. You can correct this by running:

```
>extattr +p /opt/tectia/sbin/sshd2
```

10.2.4 Exceeding Maximum CPU Time

In some environments it is possible that CPU time is restricted per user process. This can cause the transfer of large files or long-running terminal sessions to fail.

The problem can manifest itself as the signal 29 (SIGXCPU) error, which means that the allocated CPU time per process was exceeded.

Global

The global maximum CPU time can be defined in MAXCPUTIME in the SYS1.PARMLIB(BPXPRMxx) file.

```
MAXMMAPAREA(40960)
MAXCORESIZE(4194304)
MAXASSIZE(2147483647)
MAXCPUTIME(2147483647)
```

The defined SYS1.PARMLIB(BPXPRMxx) values can be easily displayed with the console command D OMVS,O.

OMVS

The CPU time can be defined in the RACF OMVS segment with the CPUTIMEMAX option as follows:

```
OMVS INFORMATION
----------------
UID= 0000000000
HOME= /
PROGRAM= /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= 0268435456
FILEPROCMAX= NONE
PROCUSERMAX= NONE
THREADSMAX= NONE
MMAPAREAMAX= NONE
```

JCL

In JCL, the CPU time can be defined with the TIME option. Defining TIME=1440 sets unlimited CPU time.

10.2.5 Auxiliary Storage Shortage

The Tectia Server for IBM z/OS server program **sshd2** may make large demands on Auxiliary Storage.

If the server causes the total allocation of virtual storage in the system to approach the amount of Auxiliary Storage available, the system may enter an Auxiliary Storage Shortage state, which will require operator action to rectify. In this state the system will fail to start new processes.

When a client requests file statistics or file transfer the server may stage the file into memory. Staging a data set means that the server creates a Hiperspace memory file and copies the whole data set into it. Each memory file will be the size of the converted data set, up to 2 GB. A memory file is retained until a file transfer operation on it has ended or the connection is ended. The server may concurrently retain several memory files.

Clients that have a file-listing feature may request file statistics for every file in a directory.

Thus a user on a remote machine may cause an Auxiliary Storage Shortage by executing the **sftp** command **ls** in a directory that contains many files or several large files, or **get** for a very large data set or file. The user must be authenticated before entering commands.

Tectia Server for IBM z/OS allows limiting the total amount of virtual storage that is used on one client connection. The limit can be set in the `SSH_SFTP_STAGEFS_CACHE_SIZE_LIMIT` environment variable.

Users should take precautions against the occurrence of an Auxiliary Storage Shortage situation as follows:

* Have a spare page data set available.

* Be prepared to identify and cancel the process causing the shortage.

* Issue the `PAGEADD` command to make the spare page data set available to the system.

10.2.6 SSHD2 Cannot Be Started as a Started Task

Please see the correct started task format from `USER.PROCLIB(SSHD2)` (see the section called "Console").

Also verify that you have the correct environment variables set. See Section 3.3.

If the task fails, please see the output of the STDERR file defined on the started task (by default `/tmp/SSHD2-sshd2.err`).

10.2.7 File Transfer Server Log Messages with Wrong Timestamps

File transfer server sft-server-g3 writes syslog messages with incorrect timestamps.

Probably the timezone for USS processes has been set only in `/etc/profile`, which is sufficient for most USS operations, but it does not get read when **sft-server-g3** is launched. Verify that `/etc/environment` contains a correct timezone declaration.

For example, setting Eastern Standard Time with Daylight Savings Time:

```
TZ=EST5EDT,M3.2.0/2:00:00,M11.1.0/2:00:00
```

For more information on the syntax of the `TZ` variable, see the z/OS USS documentation.

Appendix A Man Pages

The following server manual pages are installed to `/opt/tectia/man`:

- ssh-certd(8): Secure Shell Certificate Validator on z/OS

- ssh_certd_config(5): configuration file format for **ssh-certd** on z/OS

- ssh-dummy-shell(1): ultimately restricted shell

- ssh-externalkeys(5): instructions on using external keys with Tectia Server for IBM z/OS

- sshd-check-conf(5): check what your configuration allows or denies based on incoming user name and/or host name

- sshd2(8): Secure Shell server daemon on z/OS

- sshd2_config(5): **sshd2** configuration file format on z/OS

- sshd2_subconfig(5): advanced configuration of **sshd2** on z/OS

- sshregex(1): describes the regular expressions (or globbing patterns) used in the `sshd2_config` configuration file

For convenience, the contents of the manual pages are also shown in the following sections.

ssh-certd

ssh-certd -- Secure Shell Certificate Validator on z/OS

Synopsis

ssh-certd [-d *debug_level_spec*] [-f *config_file*] [-o *options*] [-l *listener_path*] [-F] [-v] [-V]
[-q]

Description

ssh-certd (Secure Shell Certificate Validator) is a common process for validating certificates, used primarily by **sshd2** when validating user certificates. Without a common place for the validations, all the data needed for the validation would need to be duplicated in every process doing the validations. This would be very inefficient especially in cases where very large CRLs (certificate revocation lists) are to be used. **ssh-certd** allows CRLs and CA certificates to be loaded only once and then used for all subsequent validations.

ssh-certd is normally started at boot time from /etc/rc.local or equivalent. It opens a listener socket by default at /opt/tectia/var/run/ssh-certd-listener. The location of the listener can be changed with the **CertdListenerPath** keyword of sshd2_config.

ssh-certd can be configured using command-line options or a configuration file. Command-line options override values specified in the configuration file. **ssh-certd** reads configuration data from /opt/tectia/etc/ssh_certd_config (or the file specified with the -f option on the command line). By default, the configuration file contains only the keyword UseSSHD2ConfigFile, which instructs **ssh-certd** to read the specified **sshd2** configuration file in compatibility mode, where the configuration options of **sshd2** are silently ignored, and only the options relating to certificate or general daemon configuration are read.

Start **ssh-certd** as a started task as follows:

```
===> s sshcertd
```

The started task takes modify commands as follows:

```
===> f sshcertd,<command>
```

The following modify commands are supported:

debug *debug_level*
    Set the debug level of the running Certificate Validator.

stop
    Stop the Certificate Validator.

restart
    Restart the Certificate Validator.

version

> Query the version of the Certificate Validator.

Options

The following options are available for **ssh-certd**:

`-d debug_level_spec`

> Debug mode. The server sends verbose debug output to STDERR. This option is only intended for debugging for the server. The debugging level is either a number, or a comma-separated list of assignments of the format `ModulePattern=debug_level`, for example "`*=10,sshd2=2`". This should be the first argument on the command line.

`-f` *`configuration_file`*

> Specifies the name of the configuration file. The default is `/opt/tectia/etc/ssh_certd_config`.

> ### ⓘ Note
>
> > If this option is specified, the default configuration file is not read at all.

`-o` `'option'`

> Can be used to give options in the format used in the configuration files. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file. Comment lines are not accepted. Where applicable, egrep regex format is used.

`-l` *`listener-path`*

> Specifies the path where the server will open the listener socket.

`-F`

> Disables daemon mode. The server does not spawn a new process to the background.

`-v`

> Enables verbose mode. Displays verbose debugging messages. Equal to `-d 2`. This option can also be specified in the configuration file.

`-V`

> Displays the version string.

`-q`

> Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged. This option can also be specified in the configuration file.

Configuration File

ssh-certd reads configuration data from `/opt/tectia/etc/ssh_certd_config` (or the file specified with `-f` on the command line). The file contains keyword-value pairs, one per line. Lines starting with `'#'` and empty lines are interpreted as comments.

For the format of `ssh_certd_config`, see ssh_certd_config(5).

Files

`/opt/tectia/etc/ssh_certd_config`

> Contains configuration data for **ssh-certd**. This file should be writable by root only, but it is recommended (though not necessary) that it be world-readable. For ease of migration from older installations, `ssh_certd_config` contains by default the line `"UseSSHD2ConfigFile sshd2_config"`, which instructs **ssh-certd** to read the certificate configuration from the **sshd2** configuration file and ignore the options that are not relevant to it.

ssh_certd_config

ssh_certd_config -- configuration file format for ssh-certd on z/OS

Configuration File

ssh-certd reads configuration data from /opt/tectia/etc/ssh_certd_config (or the file specified with -f on the command line). The file contains keyword-value pairs, one per line. For a description of the configuration file format, see sshd2_config(5).

The following keywords are allowed:

CertCacheFile

This keyword specifies the name of the file where the certificates and CRLs are cached when **ssh-certd** goes down.

Cert.DODPKI

If set to yes, the certificates are required to be DoD PKI compliant. The argument must be yes or no. The default is no.

CrlAutoUpdate

The argument is of the format: {yes,no}[,update_before=seconds][,min_interval=seconds]

This keyword turns on the CRL auto-update feature. When it is on, **ssh-certd** periodically tries to download the new CRL before the old one has expired. The update_before keyword specifies how many seconds before the expiration the update takes place. The min_interval sets a limit for the maximum update frequency (default minimum interval is 30 seconds).

CrlPrefetch

The argument is of the format: seconds URL. This keyword instructs **ssh-certd** to periodically download a CRL from the specified URL. The first argument specifies how often the CRL is downloaded.

ExternalMapper

This keyword specifies an external mapper program for the preceding **Pki** keyword. When a certificate is received and is valid under the Pki block in question, the external mapper is executed and the certificate is written to its standard input. The external mapper is expected to output a newline-separated list of user names. If the user name the user is trying to log in as is found in the list, the authentication succeeds; otherwise authentication using the certificate in question fails. The ExternalMapper keyword will override all **MapFile** keywords for the current (preceding) Pki keyword. If multiple ExternalMapper keywords are specified for a Pki block, the first one is used.

ExternalMapperTimeout

This keyword specifies an external mapper timeout (in seconds) for the preceding **Pki** keyword. If the server is unable to read the full output from an external mapper in the given period, the operation will fail and the external mapper program will be terminated. The default timeout is 10 seconds. If multiple `ExternalMapperTimeout` keywords are specified for a `Pki` block, the first one is used.

HostCA

The argument is of the format: `ca-certificate[,use_expired_crls=seconds]`

This keyword specifies the CA certificate (in binary or PEM (base-64) format) to be used when authenticating users using host-based authentication. The certificate received from the client must be issued by the specified CA and must contain a correct alternate name of type DNS (FQDN). If no CA certificates are specified in the configuration file, the protocol tries to do key exchange with ordinary public keys. Otherwise certificates are preferred. Multiple CAs are permitted, but only one per `HostCA` keyword.

If the additional comma-separated keyword `use_expired_crls` is given, expired CRLs will be allowed for this CA for the specified duration after the expiration, if newer CRLs are unavailable.

> ⚠️ **Caution**
>
> This feature allows a malicious party to force the use of expired CRLs if the said party can perform a denial-of-service attack against the CRL distribution point.

HostCAEkProvider

Specifies the external key provider for accessing CA certificates that are trusted for authenticating users using host-based authentication. The value is of the format `"provider:initstring"`. Currently, the only valid value for provider on z/OS is `zos-saf`. For the format of the `initstring`, see ssh-externalkeys(5).

HostCAEkProviderNoCRLs

This keyword is similar to `HostCAEkProvider`, but disables CRL checking for the CA certificates defined by `"provider:initstring"`. This option should be used for testing purposes only. In normal operations, it is highly recommended to always use CRLs.

HostCANoCRLs

This keyword is similar to `HostCA`, but disables CRL checking for the given ca-certificate.

> ℹ️ **Note**
>
> This option should be used for testing purposes only. In normal operations, it is highly recommended to always use CRLs.

LdapServers

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be checked if the point exists. Otherwise the comma-separated server list given by option `LdapServers` is used. If intermediate CA certificates are needed in certificate validity checking, this option must be used or retrieving the certificates will fail.

MapFile

This keyword specifies a mapping file for the preceding `Pki` keyword. Multiple mapping files are permitted per one `Pki` keyword. The mapping file format is described in the section called "Mapping Files".

OCSPResponderURL

Specifies the OCSP (Online Certificate Status Protocol) Responder service address in URL format, in case OCSP should be used instead of CRLs and the certificate itself does not contain a valid Authority Info Access extension with an OCSP Responder URL. 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.

If OCSP responder is defined globally or in a certificate, it is tried first; only if it fails, traditional CRL checking is tried, and if that fails, the certificate validation returns a failure.

PidFile

Specifies the file where the process ID of **ssh-certd** is written. The default is `/opt/tectia/var/run/ssh-certd-listener.pid`.

Pki

The argument is of the format: `ca-certificate[,use_expired_crls=seconds]`

This keyword enables user authentication using certificates. `ca-certificate` must be a path to a X.509 certificate in binary format. This keyword must be followed by one or more `MapFile` keywords.

The validity of a received certificate is checked separately using each of the defined `Pki` keywords in turn until they are exhausted (in which case the authentication fails), or a positive result is achieved. If the certificate is valid, the mapping files are examined to determine whether the certificate allows the user to log in (of course, the correct signature generated by a matching private key is always required in addition to everything else).

If the additional comma-separated keyword `use_expired_crls` is given, expired CRLs will be allowed for this CA for the specified duration after the expiration, if newer CRLs are unavailable.

> ⚠️ **Caution**
>
> This feature allows a malicious party to force the use of expired CRLs if the said party can perform a denial-of-service attack against the CRL distribution point.

PkiDisableCrls

If set to `yes`, this keyword disables CRL checking for the preceding `Pki` or `PkiEkProvider` keyword. The argument must be `yes` or `no`. The default is `no` (CRL checking is on).

Note that this disables only the CRL checking and only for the specific CA certificate in the `Pki` or `PkiEkProvider` parameter. OCSP checking is still done, but the result does not matter, since the fallback, the CRL, is implicitly valid when this option is active.

> ℹ️ **Note**
>
> This option should be used for testing purposes only. In normal operations, it is highly recommended to always use CRLs.

PkiEkProvider

This keyword can be used in place of the `Pki` keyword if the trusted CA keys are accessed from SAF. The value is of the format `"provider:initstring"`. Currently, the only valid value for `provider` is `zos-saf`. For the format of the `initstring`, see ssh-externalkeys(5).

QuietMode

Nothing is logged in the system log, except fatal errors. The argument must be `yes` or `no`. The default is `no`.

RandomSeedFile

Specifies the name of the random seed file.

SocksServer

With this option, **ssh-certd** can use a SOCKS4 or SOCKS5 server to connect to an LDAP server outside a firewall when making CRL queries. You can specify whether to use SOCKS5 with the option `UseSocks5`.

The argument syntax is described in sshd2_config(5).

SysLogFacility

Gives the facility code that is used when logging messages from **ssh-certd**. The possible values are: `DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7`. The default is `AUTH`.

UseSocks5

Uses SOCKS5 instead of SOCKS4 when connecting to a remote host. Note that you have to set `SocksServer` to a meaningful value. The argument must be `yes` or `no`. The default is `no` (i.e. use SOCKS4).

UseSSHD2ConfigFile

Instructs **ssh-certd** to read another configuration file in compatibility mode (the **sshd2** configuration options are allowed, but ignored).

VerboseMode

Causes **ssh-certd** to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems. Also causes **ssh-certd** to not detach from the shell to the background.

WTORoutingCodes

Specifies the z/OS routing codes for WTO messages. Valid values for routing codes are from 1 to 128. The default is `1,11`.

Mapping Files

When certificates are used in user authentication, one or more mapping files determine whether the user can log in to an account with a certificate. The mapping file must contain one or more lines in the following format:

```
account-id keyword arguments
```

`keyword` must be one of the following: `Email`, `EmailRegex`, `Subject`, `SerialAndIssuer`, or `SubjectRegex`.

`arguments` are different for each keyword. The following list describes each variation:

`Email`

Arguments: an e-mail address in standard format.

If the certificate contains the e-mail address as an alternate name, it is good for logging in as user `account-id`.

`Subject`

Arguments: a subject name in DN notation (LDAP style).

If the name matches the one in the certificate, the certificate is good for logging in as user `account-id`.

`SerialAndIssuer`

Arguments: a number and an issuer name in DN notation (LDAP style), separated by a whitespace.

If the issuer name and serial number match those in the certificate, the certificate is good for logging in as user `account-id`.

`EmailRegex`

Arguments: a regular expression (regex syntax: `ssh`, see the section called "Regex syntax: `ssh`").

If it matches an alternate name (of the type `Email`) in the certificate, the certificate is good for logging in as user `account-id`. As a special feature, if `account-id` contains a string `%subst%`, it is replaced by the first parenthesized sub-string of the regular expression before comparing it with the account the user is trying to log into.

Note that by default, the whole email address or subject name does not need to be matched. The regular expression needs only to match to some part of the string to be matched. If the whole string is required to match, the regular expression should start with `$` and end with `^`, these match the start of the line and the end of the line, respectively.

`SubjectRegex`

Works identically to `EmailRegex`, except that it matches the regular expression to the canonical subject name in the received certificate.

Empty lines and lines beginning with '#' are ignored.

Example Mapping File

```
guest Email guest@ssh.com
guest Subject C=FI, O=Example Ltd., CN=Guest User
guest SerialAndIssuer 123 C=FI, O=Foo\, Ltd., CN=Test CA
%subst% EmailRegex ([a-z]+)@example\.com
%subst% SubjectRegex C=FI, O=Example, CN=([a-z]+)
```

The example `EmailRegex` permits in users with e-mail addresses with domain `example.com` and usernames that contain only letters, each user to the account that corresponds to the user name part of the e-mail address.

The example `SubjectRegex` lets in all users with fields `C=FI` and `O=Example` in the subject name if their `CN` field contains only letters and is the account name they are trying to log into.

Note also that all characters interpreted by the regular expression parser as special characters must be escaped with a backslash if they are a part of the subject name itself. This also means that the backslash in the `SerialAndIssuer` example would have to be escaped with another backslash if the same subject name was used in a `SubjectRegex` rule.

ssh-dummy-shell

ssh-dummy-shell -- Ultimately restricted shell

Synopsis

`ssh-dummy-shell` [`-c sftp-shell`]

Description

ssh-dummy-shell is used to provide access to systems where only file transfer functionality is permitted. Users with file-transfer-only access can have **ssh-dummy-shell** as their user shell. When executed without any parameters, the program waits for the user to press any key and exit. The only way to execute programs with **ssh-dummy-shell** is to give them as command-line parameters with the `-c` option. Even then, **sft-server-g3** is the only allowed command.

Options

`-c command`

   The parameter is executed as a shell command. Only the **sftp-server** command is allowed. Any other command causes **ssh-dummy-shell** to exit immediately.

Return Status

ssh-dummy-shell returns the return value of the given command. If no command is given, 0 will be returned on exit.

Files

`/opt/tectia/etc/ssh_dummy_shell.out`

   Contains the message that is shown to the user when **ssh-dummy-shell** is executed without any parameters.

ssh-externalkeys

ssh-externalkeys -- Using external keys with Tectia Server for IBM z/OS

Description

This document contains general information about using external keys with Tectia Server for IBM z/OS.

Using External Keys

For applications capable of using external keys, two strings need to be specified: the provider name and the initialization string for the provider. These strings can be given on the command line or in a configuration file, depending on the application. The following section describes the different providers available in more detail.

The provider name and/or the initialization string may be defined in the following configuration attributes and keywords:

In `ssh-broker-config.xml`:
```
cert-validation/key-store[@type="provider",init="initstring"]
known-hosts/key-store[@type="provider",init="initstring"]
key-stores/key-store[@type="provider",init="initstring"]
```

In `sshd2_config`:
```
AuthorizationEkProvider="provider:initstring"
HostKeyEkInitString="initstring"
HostKeyEkProvider="provider"
KnownHostsEkProvider="provider:initstring"
```

In `ssh_certd_config`:
```
HostCAEkProvider="provider:initstring"
HostCAEkProviderNoCRLs="provider:initstring"
PkiEkProvider="provider:initstring"
```

External Key Providers

`zos-saf`

The zos-saf provider is used for accessing keys stored in the IBM z/OS System Authorization Facility (SAF).

The initialization string for the zos-saf provider specifies the key(s) to be used and it has the following components:

```
{KEYS([ID(xxx)]RING(xxx) [LABEL(xxx)|DEFAULT])}...
```

`KEYS(..)` may repeat. The sub-attributes are:

- `ID` - A SAF user ID signifying the owner of the key ring. If missing, the current user's ID is used.

- `RING` - Key ring name. Mandatory.

- `LABEL` - The SAF key label. If missing, and `DEFAULT` is missing, use all the keys in the key ring.

- `DEFAULT` - Use the key that is marked as the default key on the key ring. Do not specify together with `LABEL`.

Values must be written in single quotation marks if they contain single quotation marks or parenthesis.

The initialization string specified with the `HostKeyEkInitString` keyword of `sshd2_config` must point to a single private key. If the key ring contains several keys, `LABEL` must be used to distinguish between the keys.

When using a trusted key provider and the Tectia Certificate Validator, specify `KEYS` variables that include all the CA certificates needed, for example:

```
PkiEkProvider="zos-saf"
PkiEkInitString="KEYS(RING(Trusted.CAs) LABEL('Primary CA'))
KEYS(ID(SSHTEST) RING(Internal.CAs))"
```

The `key-store[@init]` attribute of `ssh-broker-config.xml` and the `AuthorizationEkProvider` keyword of `sshd2_config` can contain special strings in the key specification that are mapped according the following list:

- `%U` = user name

- `%IU` = user ID

- `%IG` = user group ID

- `%UU` = user name in upper case (*AuthorizationEkProvider only*)

- `%UL` = user name in lower case (*AuthorizationEkProvider only*)

sshd-check-conf

sshd-check-conf -- checks what your configuration allows or denies based on the incoming user name and/or host name

Synopsis

sshd-check-conf [-d *debug_level*] [-v] [-V] [-h] [-f *config_file*] [[*user@*]*host*...]

Description

sshd-check-conf checks how **sshd2** will react to an incoming user, based on the user name and the remote host name given as parameters. Currently, the parameters `AllowHosts`, `DenyHosts`, `AllowSHosts`, `DenySHosts`, `AllowUsers`, `DenyUsers`, `AllowGroups`, `DenyGroups`, `ChRootUsers`, `ChRootGroups`, `AllowTcpForwarding-ForUsers`, `DenyTcpForwardingForUsers`, `AllowTcpForwardingForGroups`, and `DenyTcpForwardingForGroups` are checked.

Options

The following options are available:

-d *debug_level_spec*

    Debug mode. The debugging level is either a number or a comma-separated list of assignments of the format `ModulePattern=`*`debug_level`*, for example "`*=10,sshd2=2`".

-v

    Enables verbose mode. Displays verbose debugging messages. Equivalent to `-d 2`.

-V

    Displays version string.

-h

    Displays a short help on command-line options.

-f *configuration_file*

    Specifies the name of the configuration file. The default is `/opt/tectia/etc/sshd2_config` or `$HOME/.ssh2/sshd2_config`, depending on who is running the program, root or normal user.

Behavior

Any non-options given on the command line will be regarded as `[user@]host` patterns (that is, the user part is optional). If the host part is a valid IP address, it is looked up from DNS. Otherwise it is interpreted as a host name and the corresponding IP addresses will be queried from DNS.

You can specify multiple patterns on the command line.

If no patterns are specified on the command line, **sshd-check-conf** will go into interactive mode where the patterns can be given one at a time and they will be checked.

You may also specify one command in interactive mode, "**dump**". This command dumps the configuration (with subconfigurations amended) for the previous pattern.

Examples

```
% sshd-check-conf -f /opt/tectia/etc/sshd2_config testuser@example.com
```

```
% sshd-check-conf -f /opt/tectia/etc/sshd2_config user1@example.org user2@example.com
```

```
% sshd-check-conf
```

sshd2

sshd2 -- Secure Shell server daemon on z/OS

Synopsis

sshd2 [-d *debug_level_spec*] [-D *debug_level_spec*] [-f *config_file*] [-h *host_key_file*] [-o *options*] [-4|-6] [-p *port*] [-F] [-v] [-V] [-g *login_grace_time*] [-i] [-q]

Description

sshd2 (Secure Shell Daemon) is the server counterpart of **sshg3**. Together, these two programs replace and extend the services rlogin and rsh, and provide secure encrypted communication channels between two hosts connected over an unsecured network. They are intended to be as easy to install and use as possible.

sshd2 can be started either using a started task or manually from USS. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

sshd2 can be configured using command-line options or a configuration file. Command-line options override values specified in the configuration file. **sshd2** reads configuration data from /opt/tectia/etc/sshd2_config (or the file specified with -f on the command line).

Subconfiguration files can also be specified in the main configuration file.

Start **sshd2** as a started task as follows:

```
===> s SSHD2
```

The started task takes modify commands as follows:

```
===> f SSHD2,<command>
```

The following modify commands are supported:

debug *debug_level*

    Set the debug level of the running server.

stop

    Stop the server. Existing connections are killed.

restart

    Restart the server. Existing connections will stay open until they are disconnected. If you have made configuration changes, existing connections will continue to use the old configuration settings while new connections will use the reconfigured settings.

    Options:

```
force
```
      Kill all existing connections and restart the server.

version

      Query the version of the server.

The **sshd2** command-line options (described in detail in the following section) can be given as an OPTS parameter on the **sshd2** "start". For example, configuration file options are entered in the following format (where *keyword* is a configuration file keyword):

```
===> S SSHD2,OPTS='-o<keyword>=<value>'
```

Under USS you can start **sshd2** manually as the SSHD2 user with the following command:

```
> /opt/tectia/sbin/sshd2
```

Options

The following options are available:

`-d` *debug_level_spec*

      Runs the server in one-shot debug mode. The server process accepts only one connection and exits after the connection has disconnected. The server sends verbose debug output to STDERR. The debugging level is either a number, or a comma-separated list of assignments of the format ModulePattern=debug_level, for example "*=10,sshd2=2". This should be the first argument on the command line.

`-D` *debug_level_spec*

      Runs the server in continuous debug mode. As `-d`, but the server accepts several connections and needs to be stopped manually when you want to finish the debugging.

`-f` *configuration_file*

      Specifies the name of the configuration file. The default is /opt/tectia/etc/sshd2_config.

> 🛈 **Note**
>
>   If this option is specified, the default configuration file is not read at all.

`-h` *host_key_file*

      Specifies the file from which the host key is read (default /opt/tectia/etc/hostkey).

`-o` *'option'*

      Can be used to give options in the format used in the configuration files. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file. Comment lines are not accepted. Where applicable, the egrep regex format is used.

-4

Specifies that **sshd2** will only use IPv4 addresses. IPv6 addresses from **ListenAddress** configuration statements, name resolution and port forwarding will be ignored.

-6

Specifies that sshd2 will only use IPv6 addresses. IPv4 addresses from **ListenAddress** configuration statements, name resolution and port forwarding will be ignored.

-p *port*

Specifies the port on which the server listens for connections. Note that when -p is specified, the server will ignore any defined **ListenAddress** and **AddressFamily** configuration statements and listen on all available IPv4 and IPv6 interfaces. The -p option is intended for testing - for greater control use the sshd2_config configuration file. The value of the **Port** configuration variable can be specified on the command line with the -o option. However, explicit port numbers on ListenAddress statements will override -oPort.

-F

Disables daemon mode. The server does not spawn a new process to the background.

-v

Enables verbose mode. Displays verbose debugging messages. Equivalent to -d 2. This option can also be specified in the configuration file with the keyword **VerboseMode**.

-V

Displays version string.

-q

Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged. This option can also be specified in the configuration file with the keyword **QuietMode**.

-g *login_grace_time*

Gives the grace time for clients to authenticate themselves (the default is 600 seconds). If the client fails to authenticate the user within this many seconds, the server disconnects and exits. The value zero indicates no limit.

-i

Specifies that **sshd2** is invoked via **inetd**.

Configuration File

sshd2 reads configuration data from /opt/tectia/etc/sshd2_config (or the file specified with -f on the command line). The file contains keyword-value pairs, one per line. Lines starting with '#' and empty lines are interpreted as comments.

For the format of sshd2_config, see sshd2_config(5).

Login Process

When a user logs in successfully, **sshd2** does the following:

1. Changes process to run with normal user privileges.

2. Sets up the basic environment.

3. Changes to the user's home directory.

4. Runs the user's shell or command.

Files

`/opt/tectia/etc/sshd2_config`

Contains configuration data for **sshd2**. This file should be writable by `root` only, but it is recommended (though not necessary) that it be world-readable.

`/opt/tectia/etc/hostkey`

Contains the private part of the host key. This file is normally created automatically by `"make install"`, but can also be created manually using **ssh-keygen-g3**. This file contains vital cryptographic information and may only be read or modified by `root`.

`/opt/tectia/etc/hostkey.pub`

Contains the public part of the host key. This file is normally created automatically by `"make install"`, but can also be created manually. This file should be world-readable but must not be writable by anyone but `root`. If it does not match the private counterpart, clients probably get confused about the server's identity.

`/opt/tectia/etc/random_seed`

This file contains a seed for the random number generator. This file must not be accessible by anyone but `root`.

`$HOME/.ssh2/authorization`

Contains information on how the server will verify the identity of a user. For more information, see Section 5.6.2.

`$HOME/.hushlogin`

If this file exists, **sshd2** will not print information during login. (This is normally the user's last login time, message of the day and mail check.)

`/etc/nologin`

If this file exists, **sshd2** refuses to let anyone except `root` log in. The contents of the file are displayed to anyone trying to log in. The file should be world-readable.

/etc/nologin_<*hostname*>

As above, but the file name is constructed from the name of the host. Check the output of hostname to see which name you should use in the file name. This functionality is supposed to be used by clustered machines (which share /etc).

$HOME/.rhosts

This file contains host-username pairs, separated by a white space, one per line. The given user is permitted to log in from the given host without a password. The same file is used by **rlogind** and **rshd**. **sshd2** differs from **rlogind** and **rshd** in that it requires public host-key authentication from the ssh2 server running on this host in addition to validating the host name retrieved from domain name servers. The file must be writable only by the user; it is recommended that it is not accessible by others.

It is also possible to use netgroups in the file. Either host or user name may be of the form +@groupname to specify all hosts or all users in the group.

$HOME/.shosts

For **sshg3**, this file is exactly the same as for .rhosts. However, this file is not used by **rlogin** and **rshd**, so using this permits access using **sshg3** only.

/etc/hosts.equiv

This file is used during .rhosts authentication. In its simplest form, this file contains host names, one per line. Users on those hosts are permitted to log in without a password, provided they have the same user name on both machines. The host name may also be followed by a user name; such users are permitted to log in as any user on this machine (except root). Additionally, the syntax +@group can be used to specify netgroups. Negated entries start with '-'.

If the client host/user is successfully matched in this file, login is automatically permitted provided the client and server user names are the same. Additionally, successful host-based authentication is normally required. This file must be writable only by root; it is recommended that it be world-readable.

> ⚠ **Caution**
>
> It is almost never a good idea to use user names in hosts.equiv. Note that it really means that the named user(s) can log in as anybody, including bin, daemon, adm, and other accounts that own critical binaries and directories. Using a user name practically grants the user root access. The only valid use for user names should be in negative entries. Note that this warning also applies to **rsh**/**rlogin**.

/etc/shosts.equiv

This is processed exactly as /etc/hosts.equiv. However, this file is not used by **rlogin** and **rshd**, so using this permits access using **sshg3** only.

$HOME/.ssh2/knownhosts/*xxxxyyyy*.pub

These are the public host keys of hosts that a user wants to log in from using host-based authentication (equivalent to RhostsRSAAuthentication in ssh1). Also, users have to set up their $HOME/.shosts

(only used by ssh) or `$HOME/.rhosts` files (insecure, as it is used by the **r*** commands also). If the user name is the same on both hosts, it is adequate to put the public host key to `/opt/tectia/etc/knownhosts` and add the host name to `/etc/shosts.equiv` (or `/etc/hosts.equiv`).

xxxx denotes the host name (FQDN) and *yyyy* denotes the public-key algorithm of the key.

For example, if zappa.foo.fi's host-key algorithm is ssh-dss, the host key is contained in the file `zappa.foo.fi.ssh-dss.pub` in the `knownhosts` directory.

Possible names for public-key algorithms are `ssh-dss` and `ssh-rsa`.

`/opt/tectia/etc/knownhosts/`*xxxxyyyy*`.pub`

As above, but system-wide. These settings can be overridden by the user by putting a file with the same name to the `$HOME/.ssh2/knownhosts` directory.

sshd2_config

sshd2_config -- configuration file format for sshd2 on z/OS

Configuration File

sshd2 reads configuration data from `/opt/tectia/etc/sshd2_config` (or the file specified with `-f` on the command line). The file contains keyword-value pairs, one per line.

A configuration file can begin with "metaconfiguration" information, that is, information configuring the configuration language itself.

If the configuration file starts with a line matching the following egrep-style regular expression:

```
#.*VERSION[ \t\f]+[0-9]+.[0-9]+
```

it is interpreted as the version of the configuration style. If this kind of line is not found, the version is considered to be "1.0".

The version string can be followed by one or more metaconfiguration parameters. The lines have to start with '#', and they have to match the following egrep-style regular expression:

```
#[# \t]+[A-Z0-9]+[ \t]+.*
```

The parsing of metaconfiguration directives stops with the first non-recognized line.

Version 1.1 and later recognize the following parameter:

REGEX-SYNTAX

    This denotes the regex (regular expression) syntax used to parse the configuration file in question. The regex syntax is used in parsing the labels, lists, and so on, and when matching something with the regex patterns specified in the configuration file.

    The value can be `egrep`, `ssh`, `zsh_fileglob` or `traditional` (the arguments are not case-sensitive). `zsh_fileglob` and `traditional` are synonymous.

Subconfiguration files can be specified in the main configuration file, see **HostSpecificConfig** and **User-SpecificConfig**.

> ℹ️ **Note**
>
>   If changes are made in the main configuration file, **sshd2** will have to be restarted as instructed in Section 3.1.3.

In the configuration file, empty lines and lines starting with '#' are ignored as comments.

Otherwise a line is of the format `'keyword arguments'`. Note that it is possible to enclose arguments in quotes, and use the standard C convention.

> ### ℹ Note
>
> The keywords are not case-sensitive but the arguments are case-sensitive.

The possible keywords and their meanings are as follows :

AddressFamily

This keyword specifies the TCP/IP address families that **sshd2** should use when listening for SSH connections and connecting to tunneled ports. The value may be `inet` for IPv4, `inet6` for IPv6, or `any`, which means that **sshd2** will listen on both protocols and connect to an IPv4 or IPv6 port, whichever it finds first. The default value is `inet`.

AllowAgentForwarding or **ForwardAgent**

Specifies whether agent forwarding is permitted. This parameter is implemented mainly for completeness. Usually, you should allow users to freely forward agent connections. The argument must be `yes` or `no`. The default is `yes`.

AllowedAuthentications

This keyword specifies the authentication methods that are allowed. Known authentication methods are: `keyboard-interactive`, `password`, `publickey`, and `hostbased`. The default is `"publickey,password"`. The order of the listed authentication methods is important, as the first method will be used that matches the methods the client is proposing.

With **RequiredAuthentications**, the system administrator can force the users to complete several authentications before they are considered authenticated.

AllowGroups

This keyword can be followed by any number of group name patterns, separated by commas. If specified, login is allowed only if one of the groups the user belongs to matches one of the patterns. Patterns are matched using the egrep syntax (see sshregex(1)), or the syntax specified in the metaconfiguration header of the configuration file. You can use the comma character (,) in the patterns by escaping it with backslash (\). By default, all groups are allowed to log in.

AllowHosts

This keyword can be followed by any number of host name patterns, separated by commas. If specified, login is allowed only from hosts whose name matches one of the patterns. Patterns are matched using the egrep syntax (see sshregex(1)), or the syntax specified in the metaconfiguration section of the configuration file. If you want the pattern to be matched with the host's IP address only (ignoring the canonical host name), prefix your pattern with `"\i"`. You can also use subnet masks (e.g 127.0.0.0/8) by prefixing

---

the pattern with `"\m"`. DNS is used to map the client's host name into a canonical host name. If the name cannot be mapped, the IP address is used as the host name. By default, all hosts are allowed to connect.

AllowSHosts

This keyword can be followed by any number of host name patterns, separated by commas, same as the option `AllowHosts`. The entries in `.shosts`, `.rhosts`, `/etc/hosts.equiv` and `/etc/shosts.equiv` are ignored if they do not match one of the patterns given here (if there are any).

AllowTcpForwarding

Specifies whether TCP forwarding is permitted.

> ℹ **Note**
>
> Disabling TCP forwarding does not improve security at all unless you deny the user shell access at the same time (see ssh-dummy-shell(1)): users that have a shell can always install their own forwarders.

The argument must be `yes` or `no`. The default is `yes`.

AllowTcpForwardingForGroups

The syntax is the same as in `AllowGroups`, but instead of login, this controls the ability to forward ports in remote or local forwarding. See the security note under option **AllowTcpForwarding**.

AllowTcpForwardingForUsers

The syntax is the same as in `AllowUsers`, but instead of login, this controls the ability to forward ports in remote or local forwarding. See the security note under option **AllowTcpForwarding**.

AllowUsers

This option can be followed by any number of patterns of the form `user` or `user@host`, separated by commas. The details explained under option **AllowHosts** apply accordingly. By default, all users are allowed to log in.

Note that all the other login authentication steps must still be successfully completed. `AllowUsers` and **DenyUsers** are additional restrictions.

AuthHostbased.Cert.Required

This keyword specifies whether the client must present a host certificate during user authentication. If this option is set to `yes`, also the `AllowedAuthentications` keyword must contain the value `hostbased`. If the server does not receive a certificate, the authentication fails. The argument must be `yes` or `no`. The default is `no`.

AuthHostbased.Cert.ValidationMethods

This keyword specifies the method used for certificate validation during host-based user authentication. Its value can be `tectia` or `saf`, or both (`saf,tectia`). The default is `tectia`.

If `saf` is specified, RACF/SAF is used for validating client host certificates. The host certificates must exist in a trusted key ring defined by the **`KnownHostsEkProvider`** keyword. Note that when only SAF validation is used, the certificate validity period and revocation status are not checked.

If `tectia` is specified (or the keyword is missing from the configuration), the Tectia Certificate Validator (**ssh-certd**) is used for validating client host certificates. The host certificates must be issued by a trusted certification authority defined in the **`HostCA`**, **`HostCANoCRLs`**, **`HostCAEkProvider`**, or **`HostCAEkProvider-NoCRLs`** keyword of `ssh_certd_config`.

If both values are specified (`saf,tectia`), the RACF/SAF validation is performed first and after that the Tectia validation. The host certificates must exist in a trusted key ring defined by the `KnownHostsEkProvider` keyword. Also the CA certificate of the issuing certification authority must exist in a trusted key ring defined by the `HostCAEkProvider` or `HostCAEkProviderNoCRLs` keyword of `ssh_certd_config`.

AuthInteractiveFailureTimeout

Specifies the delay in seconds of the server after a failed attempt to log in using keyboard-interactive and password authentication. The default is 2.

AuthKbdInt.NumOptional

Specifies how many optional submethods must be passed before the authentication is considered a success (note that all required submethods must always be passed). See `AuthKbdInt.Optional` for specifying optional submethods, and `AuthKbdInt.Required` for required submethods. The default is 0, although if no required submethods are specified, the client must always pass at least one optional submethod.

AuthKbdInt.Optional

Specifies the optional submethods keyboard-interactive will use. Currently only the password and plugin submethods are supported on z/OS. `AuthKbdInt.NumOptional` specifies how many optional submethods must be passed. The `keyboard-interactive` authentication method is considered a success when the specified number of optional submethods and all required submethods are passed. The `plugin` submethod is special. It can be used if a system administrator wants to create a new authentication method. See also `AuthKbdInt.NumOptional` and `AuthKbdInt.Required`.

AuthKbdInt.Plugin

Specify this to point to a program that is used by the `plugin` submethod in `keyboard-interactive`. **sshd2** converses with this program using a line-based protocol, so it is easy to implement it, for example as a shell script. If the `plugin` submethod is used, and this is not set, or the specified program does not exist or cannot be run, the submethod will fail, which may cause the whole user authentication to fail. This is not set by default. More information about the protocol can be found in the distribution package.

`RFC.kbdint_plugin_protocol` has a description of the protocol used and an example script is called `kbdint_plugin_example.sh`. Note that the program is run with the privileges of the **sshd2** process, typically root, so be careful.

AuthKbdInt.Required

Specifies the required submethods that must be passed before the `keyboard-interactive` authentication method can succeed. See `AuthKbdInt.Optional`.

AuthKbdInt.Retries

Specifies how many times the user can retry `keyboard-interactive`. The default is 3.

AuthorizationEkProvider

This keyword specifies the external key provider for accessing external public keys and certificates used for user public-key authentication. The value is of the format `"provider:initstring"`. Currently, the only valid value for provider on z/OS is `zos-saf`. For the format of the `initstring`, see ssh-externalkeys(5).

AuthorizationEkInitStringMapper

This keyword specifies a path to an external program (for example, `"/path/to/mapper.sh"`) that can be used for generating the initialization string used in public-key authentication with external keys. The usage of the mapper program should be the following:

```
mapper.sh username "initstring"
```

`username` is the name of the user found in the certificate, and `initstring` is the initialization string as it is given in the `AuthorizationEkProvider` keyword. The program should print the modified initialization string into standard output.

AuthorizationEkInitStringMapperTimeout

This keyword specifies a timeout (in seconds) for the `AuthorizationEkInitStringMapper`. If the server is unable to read the full output from the mapper in the given period, the operation will fail and the mapper program will be terminated. The default timeout is 10 seconds.

AuthorizationFile

Specifies the name of the user's authorization file. The default is `$HOME/.ssh2/authorization`.

AuthorizedKeysFile

Specifies the name of the user's authorized keys file. This is given as a pattern string which is expanded by **sshd2**.

`%D` is the user's home directory,
`%U` is the user's login name,

%UU is the user's login name in upper case,

%UL is the user's login name in lower case,

%IU is the user's user ID (uid),

%IG is the user's group ID (gid).

The file is a legacy-format file containing multiple public keys so that each line holds a single public key. Keys are in the ssh1/openssh public key format.

This option is disabled by default.

The public-key option `from="host"` is equivalent to `allow-from="host"` and `from="!host"` is equivalent to `deny-from="host"`. Also the `$SSH_ORIGINAL_COMMAND` is equivalent to `$SSH2_ORIGINAL_COMMAND`. Note that the public-key option `permitopen="host:port"` is not supported, please see the **ForwardACL** configuration option to achieve a similar setup.

AuthPassword.ChangePlugin

Set this to the path of the password change plug-in, typically `ssh-passwd-plugin` (if you have the binary packages or you have configured the source with `--with-passwd-plugin`). This allows the password to be changed during the authentication phase, instead of using a system's **passwd** command to do it. This replaces the actual session, requiring the user to log in again. This option is also used by the `password` submethod of `keyboard-interactive`. By default this is not set.

AuthPublicKey.Algorithms

Specifies the public key signature algorithms to be used in client authentication.

The supported algorithms are `rsa-sha2-256`, `rsa-sha2-512`, `ssh-dss`, `ssh-dss-sha224@ssh.com`, `ssh-dss-sha256@ssh.com`, `ssh-dss-sha384@ssh.com`, `ssh-dss-sha512@ssh.com`, `x509v3-sign-dss`, `x509v3-sign-dss-sha224@ssh.com`, `x509v3-sign-dss-sha256@ssh.com`, `x509v3-sign-dss-sha384@ssh.com`, `x509v3-sign-dss-sha512@ssh.com`, `ssh-rsa`, `ssh-rsa-sha224@ssh.com`, `ssh-rsa-sha256@ssh.com`, `ssh-rsa-sha384@ssh.com`, `ssh-rsa-sha512@ssh.com`, `x509v3-sign-rsa`, `x509v3-sign-rsa-sha224@ssh.com`, `x509v3-sign-rsa-sha256@ssh.com`, `x509v3-sign-rsa-sha384@ssh.com`, `x509v3-sign-rsa-sha512@ssh.com`, `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, `ecdsa-sha2-nistp521`, `x509v3-ecdsa-sha2-nistp256`, `x509v3-ecdsa-sha2-nistp384`, and `x509v3-ecdsa-sha2-nistp521`.

Multiple public key signature algorithms can be specified as a comma-separated list. Special values for this option are:

- `Any` - includes all supported algorithms.

- `AnyStd` - includes algorithms from the IETF SSH standards (`ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, `ecdsa-sha2-nistp521`, `x509v3-ecdsa-sha2-nistp256`, `x509v3-ecdsa-sha2-nistp384`, `x509v3-ecdsa-sha2-nistp521`, `x509v3-sign-dss`, `x509v3-sign-rsa`, `ssh-dss`, `ssh-rsa`).

- `AnyPublicKeyAlgorithm` - the same as `Any`.

- `AnyStdPublicKeyAlgorithm` - the same as `AnyStd`.

The default list is: `rsa-sha2-256`, `rsa-sha2-512`, `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, `ecdsa-sha2-nistp521`, `x509v3-ecdsa-sha2-nistp256`, `x509v3-ecdsa-sha2-nistp384`, `x509v3-ecdsa-sha2-nistp521`, `ssh-rsa`, `ssh-rsa-sha256@ssh.com`, `ssh-dss`, `ssh-dss-sha256@ssh.com`, `x509v3-sign-dss`, `x509v3-sign-dss-sha256@ssh.com`, `x509v3-sign-rsa`, and `x509v3-sign-rsa-sha256@ssh.com`.

AuthPublicKey.Cert.Required

This keyword specifies whether the client must present a user certificate during user authentication. If this option is set to `yes`, also the **AllowedAuthentications** keyword must contain the value `publickey`. If the server does not receive a certificate, the authentication fails. The argument must be `yes` or `no`. The default is `no`.

AuthPublicKey.Cert.ValidationMethods

This keyword specifies the method used for certificate validation during user public-key authentication. Its value can be `tectia` or `saf`, or both (`saf,tectia`). The default is `tectia`.

If `saf` is specified, RACF/SAF is used for validating user certificates. The user certificates must exist in a trusted key ring defined by the **AuthorizationEkProvider** keyword. Note that when only SAF validation is used, the certificate validity period and revocation status are not checked.

If `tectia` is specified (or the keyword is missing from the configuration), the Tectia Certificate Validator (**ssh-certd**) is used for validating user certificates. The user certificates must be issued by a trusted certification authority defined in the **Pki** or **PkiEkProvider** keyword of `ssh_certd_config`.

If both values are specified, the user certificate is read from RACF/SAF and the user name is taken form the certificate. After that the Tectia validation is performed. The CA certificate of the issuing certification authority must exist in a trusted key ring defined by the **PkiEkProvider** keyword of `ssh_certd_config`.

AuthPublicKey.MaxSize

Specifies the maximum size of a public key that can be used to log in. Value `0` disables the check. The default is 0 (disabled).

AuthPublicKey.MinSize

Specifies the minimum size of a public key that can be used to log in. Value `0` disables the check. The default is `0` (disabled).

BannerMessageFile

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

The default is `/opt/tectia/etc/ssh_banner_message` (if defined).

CertdListenerPath

Specifies the path where the server tries to connect to the Certificate Validator. Mainly intended for debugging and testing. The default is `/opt/tectia/var/run/ssh-certd-listener`.

CheckMail

Makes **sshd2** print information on whether there is new mail or not when a user logs in interactively. (On some systems this information is also printed by the shell, `/etc/profile`, or equivalent.) The argument must be `yes` or `no`. The default is `yes`.

ChRootGroups

This option works like `ChRootUsers`, except that it can be used to list groups instead of single users. Groups are listed on the server in `/etc/group`. Follows the logic of **DenyGroups**.

ChRootUsers

sshd2 gives all users listed here a chrooted environment (e.g. `/home`). This stops users from accessing sensitive information on the server's file system. Users are defined on the server in `/etc/group`. More than one group can be listed, separated by a comma. This is not a default option. The logic follows that of **DenyUsers**.

Ciphers

Specifies the ciphers to use for encrypting the session. The supported ciphers are `aes128-ctr`, `aes192-ctr`, `aes256-ctr`, `aes128-gcm@openssh.com`, `aes256-gcm@openssh.com`, `aes128-cbc`, `aes192-cbc`, `aes256-cbc`, `3des-cbc`, `arcfour`, `blowfish-cbc`, `cast128-cbc`, `twofish128-cbc`, `twofish192-cbc`, `twofish256-cbc`, `twofish-cbc`, `cast128-12-cbc@ssh.com`, `seed-cbc@ssh.com`, and `rijndael-cbc@ssh.com`.

Multiple ciphers can be specified as a comma-separated list. Special values for this option are:

- `Any` - includes all supported ciphers plus `none`

- `AnyStd` - includes ciphers from the IETF SSH standards (`aes128-ctr`, `aes192-ctr`, `aes256-ctr`, `aes128-gcm@openssh.com`, `aes256-gcm@openssh.com`, `aes128-cbc`, `aes192-cbc`, `aes256-cbc`, `3des-cbc`, `arcfour`, `blowfish-cbc`, `cast128-cbc`, `twofish128-cbc`, `twofish192-cbc`, `twofish256-cbc`, `twofish-cbc`) plus `none`.

- `none` - no encryption; the data on the line is in plaintext.

- `AnyCipher` - the same as `Any`, but excludes `none`.

- `AnyStdCipher` - the same as `AnyStd`, but excludes `none`.

The default list is `aes128-ctr`, `aes192-ctr`, `aes256-ctr`, `aes128-gcm@openssh.com`, `aes256-gcm@openssh.com`, `aes128-cbc`, `aes192-cbc`, `aes256-cbc` and `3des-cbc`.

DenyGroups

This keyword can be followed by any number of group name patterns, separated by commas. If specified, login is denied if one of the groups the user belongs to matches one of the patterns. Otherwise, this option is parsed and matched identically to `AllowGroups`. By default, all users are allowed to log in.

If a user's group matches a pattern in both `DenyGroups` and `AllowGroups`, login is denied.

Note that all other authentication steps must still be successfully completed. `AllowGroups` and `DenyGroups` are additional restrictions and never increase the tolerance.

DenyHosts

This keyword can be followed by any number of host name patterns, separated by commas. If specified, login is denied from hosts whose names match any of the patterns. See `AllowHosts`.

DenySHosts

This keyword can be followed by any number of host name patterns, separated by commas, just as the option `DenyHosts`. The entries in `.shosts`, `.rhosts`, `/etc/hosts.equiv` and `/etc/shosts.equiv` are ignored if they match one of the patterns given here (if there are any). See `AllowSHosts`.

DenyTcpForwardingForGroups

The syntax is the same as in `DenyGroups`, but instead of login, this controls the ability to forward ports, in remote or local forwarding. See the security note under option `AllowTcpForwarding`.

DenyTcpForwardingForUsers

The syntax is the same as in `DenyUsers`, but instead of login, this controls the ability to forward ports, in remote or local forwarding. See the security note under option `AllowTcpForwarding`.

DenyUsers

This is the opposite of `AllowUsers` and works accordingly. If a user's name matches a pattern in both `DenyUsers` and `AllowUsers`, login is denied.

Note that all other authentication steps must still be successfully completed. `AllowUsers` and `DenyUsers` are additional restrictions.

DisableVersionFallback

Selects whether to disable fallback compatibility code for earlier, or otherwise incompatible versions of software. Do not disable this unless you know what you are doing. The argument must be `yes` or `no`. The default is `no`.

ExternalAuthorizationProgram

If set, this program is run to verify whether the user is authorized to log in. **sshd2** converses with this program using a line-based protocol, so it is easy to implement for example as a shell script. If this is set, and the program does not exist or cannot be run, authorization (user login) will be denied. This will not be set by default. More information about the protocol can be found in the distribution package, `RFC.authorization_program_protocol` has a description of the protocol used and an example script is called `ext_authorization_example.sh`. Note that the program is run with the privileges of the **sshd2** process, typically root, so be careful.

ForwardACL

With this option, you can have more fine-grained control over what the client is allowed to forward and where. The format for this option is

`(allow|deny) (local|remote) user-pat forward-pat [originator-pat]`

`user-pat` will be used to match the client user, as specified under the option **UserSpecificConfig**.

`forward-pat` is a pattern of format `host-id[%port]`. This has different interpretations depending on whether the ACL is specified for local or remote forwardings. For local forwardings, `host-id` will match with target host of the forwarding, as specified under option **AllowHosts**. `port` will match the target port. Also, if the client sent a host name, the IP is looked up from the DNS, which will be used to match the pattern. For remote forwardings, where the forwarding target is not known (the client handles that end of the connection), this will be used to match the listen address specified by the user (and as such is not as usable as with local forwardings). `port` will match the port the server is supposed to be listening to with this forwarding.

With local forwards, `originator-pat` will match the originator address that the client has reported. Remember, if you do not administer the client machine, or the users on that machine have shell access, they may use a modified copy of **ssh** that can be used to lie about the originator address. Also, with NATs (Network Address Translation) the originator address will not be meaningful (it will probably be an internal network address). Therefore you should not rely on the originator address with local forwardings, unless you know exactly what you are doing. With remote forwardings, on the other hand, `originator-pat` will match with the IP address of the host connecting to the forwarded port. This will be valid information, as it is the server checking the information.

If you specify any `allow` directives, all forwardings in that class (local or remote) not specifically allowed will be denied. (Note that local and remote forwardings are separate in this respect; e.g. if you have one "allow remote" definition, local forwardings are still allowed, pending other restrictions.) If a forwarding matches both `allow` and `deny` directives, the forwarding will be denied. Also, if you have specified any of the options `{Allow,Deny}TcpForwardingFor{Users,Groups}` or `AllowTcpForwarding`, and the forwarding for the user is disabled with those, an `allow` directive will not re-enable the forwarding for the user. Forwarding is enabled by default.

ForwardAgent

See **AllowAgentForwarding** or **ForwardAgent**.

HostbasedAuthForceClientHostnameDNSMatch

If the host name given by the client does not match the one found in DNS, fail host-based authentication. Defaults to no. Note that this differs from 2.4 and earlier releases.

HostCertificateFile

This keyword works very much like **PublicHostKeyFile**, except that the file is assumed to contain an X.509 certificate in binary format. The keyword must be paired with a corresponding HostKeyFile option. If multiple certificates with the same public-key type (DSA, ECDSA or RSA) are specified, only the first one is used.

HostIdMappingHostnames

A user certificate might contain a **HostIdMapping** field. That field is used by SAF to determine the local user name of the user. If the **HostIdMapping** field should be processed in Tectia Server for IBM z/OS, the HostIdMappingHostnames keyword can be used. It specifies the list of host names that the server recognizes. If the user certificate has the correct host name in the **HostIdMapping** host name field, the user name associated with that host name (specified in the certificate) is used. Note that if HostIdMappingHostnames is used, the Tectia certificate validation must be performed (the AuthPublicKey.Cert.ValidationMethods keyword must be set to saf,tectia).

HostKeyAlgorithms

Specifies the host key signature algorithm to be used in server authentication and host-based authentication.

The supported algorithms are rsa-sha2-256, rsa-sha2-512, ssh-dss, ssh-dss-sha224@ssh.com, ssh-dss-sha256@ssh.com, ssh-dss-sha384@ssh.com, ssh-dss-sha512@ssh.com, x509v3-sign-dss, x509v3-sign-dss-sha224@ssh.com, x509v3-sign-dss-sha256@ssh.com, x509v3-sign-dss-sha384@ssh.com, x509v3-sign-dss-sha512@ssh.com, ssh-rsa, ssh-rsa-sha224@ssh.com, ssh-rsa-sha256@ssh.com, ssh-rsa-sha384@ssh.com, ssh-rsa-sha512@ssh.com, x509v3-sign-rsa, x509v3-sign-rsa-sha224@ssh.com, x509v3-sign-rsa-sha256@ssh.com, x509v3-sign-rsa-sha384@ssh.com, x509v3-sign-rsa-sha512@ssh.com, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, x509v3-ecdsa-sha2-nistp256, x509v3-ecdsa-sha2-nistp384, and x509v3-ecdsa-sha2-nistp521.

Multiple algorithms can be specified as a comma separated list. Special values for this option are:

- Any - includes all supported host key algorithms.

- AnyStd - includes host key algorithms from the IETF SSH standards (ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, x509v3-ecdsa-sha2-nistp256, x509v3-ecdsa-sha2-nistp384, x509v3-ecdsa-sha2-nistp521, x509v3-sign-dss, x509v3-sign-rsa, ssh-dss, ssh-rsa).

- `AnyHostKeyAlgorithm` - the same as `Any`.

- `AnyStdHostKeyAlgorithm` - the same as `AnyStd`.

The default list is `ecdsa-sha2-nistp521`, `ecdsa-sha2-nistp384`, `ecdsa-sha2-nistp256`, `rsa-sha2-256`, `rsa-sha2-512`, `x509v3-ecdsa-sha2-nistp256`, `x509v3-ecdsa-sha2-nistp384`, `x509v3-ecdsa-sha2-nistp521`, `ssh-rsa`, `ssh-rsa-sha256@ssh.com`, `ssh-dss`, `ssh-dss-sha256@ssh.com`, `x509v3-sign-dss`, `x509v3-sign-dss-sha256@ssh.com`, `x509v3-sign-rsa`, and `x509v3-sign-rsa-sha256@ssh.com`.

HostKeyFile

Specifies the file containing the private host key (default `/opt/tectia/etc/hostkey`). The directory should be readable only by root.

HostKeyEkInitString

Specifies the initialization string for the external host key provider. This is ignored when the keyword `HostKeyEkProvider` is not present. The key specification must define exactly one key. It must be a personal RSA key that contains both a private key and a certificate. See ssh-externalkeys(5) for details about specifying initialization strings.

HostKeyEkProvider

Specifies the external key provider for accessing the server host key. Currently, the only valid value on z/OS is `zos-saf`. See ssh-externalkeys(5) for details about specifying providers.

HostKey.Cert.Required

This keyword specifies whether the local server must present a host certificate to the remote client during server authentication (or whether the local client must present a host certificate to a remote server during host-based user authentication).

If set to `yes`, a certificate must be defined either with the `HostCertificateFile` keyword or with the `HostKeyEkProvider` keyword.

If set to `no`, a certificate is not required. In this case, if the `HostKeyEkProvider` keyword specifies a certificate in SAF, only the public key found in the certificate is used (but `HostCertificateFile` is used normally, if defined).

If set to `optional`, both certificate and public key are used. In this case, if the `HostKeyEkProvider` keyword specifies a certificate in SAF, the certificate and the public key found in the certificate are used (but `HostCertificateFile` is used normally, if defined).

The argument must be `yes`, `no`, or `optional`. The default is `no`.

HostSpecificConfig

Specifies a subconfiguration file for **sshd2**. The syntax for this option is:

```
pattern subconfig-file
```

`pattern` is used to match the client host, as specified under option **AllowHosts**. The file `subconfig-file` is read, and configuration data is amended accordingly. The file is read before any actual protocol transactions begin, and you can specify most of the options allowed in the main configuration file. You can specify more than one subconfiguration file, in which case the patterns are matched and the files read in the specified order. Configuration option values defined later will either override or amend the previous value, depending on the option. The effect of redefining an option is described in the documentation for each option. For example, setting `Ciphers` in the subconfiguration file will override the old value, but setting `AllowUsers` will amend the value. See sshd2_subconfig(5) for more thorough documentation on what you can set in this subconfiguration file.

See also option **UserSpecificConfig**.

IdentityDispatchUsers

This keyword is used in conjunction with certificate authentication. It is followed by a user name pattern as in the **AllowUsers** keyword. If the login user name given by the client matches to the pattern, it is not used, but instead the user name is taken from the user certificate.

IdleTimeOut

Sets the idle timeout limit to time either in seconds (`s` or nothing after the number), in minutes (`m`), in hours (`h`), in days (`d`), or in weeks (`w`). If the connection has been idle (all channels) this long, the connection is closed. The default is 0 (zero), which disables idle timeouts.

IgnoreRhosts

Specifies that the `.rhosts` and `.shosts` files will not be used in host-based authentication (see **AllowedAuthentications**). `/etc/hosts.equiv` and `/etc/shosts.equiv` are still used as before. The argument must be `yes` or `no`. The default is `no`.

IgnoreRootRhosts

Specifies that the `.rhosts` and `.shosts` files will not be used in authentication for root. The default is the value of `IgnoreRhosts`.

KeepAlive

Specifies whether the system should send keepalive messages to the other side. If they are sent, a broken connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily, and this can be annoying in some situations. On the other hand,

if keepalive messages are not sent, sessions may hang indefinitely on the server, leaving "ghost" users and consuming server resources.

The default is `yes` (to send keepalives), and the server will notice if the network goes down or the client host reboots. This avoids infinitely hanging sessions.

To disable keepalives, the value should be set to `no` in both the server and the client configuration files.

KEXs

Specifies the key exchange algorithm to be used. The supported algorithms are:

```
diffie-hellman-group1-sha1
diffie-hellman-group14-sha1
diffie-hellman-group14-sha256
diffie-hellman-group16-sha512
diffie-hellman-group18-sha512
diffie-hellman-group14-sha224@ssh.com
diffie-hellman-group14-sha256@ssh.com
diffie-hellman-group15-sha256@ssh.com
diffie-hellman-group15-sha384@ssh.com
diffie-hellman-group16-sha384@ssh.com
diffie-hellman-group16-sha512@ssh.com
diffie-hellman-group18-sha512@ssh.com
diffie-hellman-group-exchange-sha1
diffie-hellman-group-exchange-sha256
ecdh-sha2-nistp256
ecdh-sha2-nistp384
ecdh-sha2-nistp521
```

Multiple KEXs can be specified as a comma-separated list. Special values for this option are:

- `Any` - includes all supported KEXs.

- `AnyStd` - includes the following KEXs from the IETF SSH standards: `diffie-hellman-group14-sha1`, `diffie-hellman-group1-sha1`, `ecdh-sha2-nistp256`, `ecdh-sha2-nistp384`, `ecdh-sha2-nistp521`.

- `AnyKEX` - the same as `Any`.

- `AnyStdKEX` - the same as `AnyStd`.

The default algorithms are `ecdh-sha2-nistp521`, `ecdh-sha2-nistp384`, `ecdh-sha2-nistp256`, `diffie-hellman-group14-sha1`, `diffie-hellman-group14-sha256`, `diffie-hellman-group16-sha512`, `diffie-hellman-group18-sha512`, `diffie-hellman-group14-sha256@ssh.com`, and `diffie-hellman-group-exchange-sha256`.

KnownHostsEkProvider

Specifies the external key provider for accessing external host public keys used for host-based user authentication. The value is of the format `"provider:initstring"`. Currently, the only valid value for `provider` on z/OS is `zos-saf`. For the format of the `initstring`, see ssh-externalkeys(5).

ListenAddress

Specifies the IP address of the interface where the **sshd2** server socket is bound. The format for this option is

```
ip-address [port] ,
```

where `port` is optional. IPv6 addresses may be enclosed in brackets and must be if they start with a colon. The port, if not defined here, will be the value of the last `Port` definition (or the default, 22, if `Port` has not been defined). If the specified IP address is `any`, **sshd2** will listen to all interfaces. If **sshd2** should listen to only some interfaces, specify the `ListenAddress` parameter for each interface.

The default is `any`.

ListenerRetryInterval

Declares how often **sshd2** tries/retries to create listeners which fail to open initially or later get deleted by the operating system.

The interval is given either in seconds (`s` or nothing after the number), in minutes (`m`), in hours (`h`), in days (`d`), or in weeks (`w`).

The default is 0 (zero), which means **sshd2** does not retry but gives up and exits immediately when a listener fails to open or dies.

LoadControl.Active

Specifies whether load control is enabled. The purpose of load control is to help keep the server running when the load is high, that is, when the number of current connections is near **MaxConnections**. The argument must be `yes` or `no`. The default is `no`.

> **ⓘ Note**
>
> If `MaxConnections` is set to `0` or `1`, load control is disabled even if `LoadControl.Active` is set to `yes`.

LoadControl.DiscardLimit

When Tectia Server for IBM z/OS's load is high, connections from IP addresses that have not recently had a successful authentication are discarded. This keyword specifies the number of connections accepted from any IP address. The allowed value range is from 1 to MaxConnections-1. The default is 90 percent of the value of **MaxConnections**.

LoadControl.WhitelistSize

Specifies the length of the white list in which Tectia Server for IBM z/OS keeps the IP addresses of connections that have had a successful authentication. The default is 1000.

LoginGraceTime

The server disconnects after this time if the user has not successfully logged in. If the value is `0`, there is no time limit. The default is 600 (seconds).

MACs

Specifies the MAC (Message Authentication Code) algorithm to use for data integrity verification. The supported algorithms are:

```
hmac-md5
hmac-md5-96
hmac-sha1
hmac-sha1-96
hmac-sha2-256
hmac-sha256-2@ssh.com
hmac-sha224@ssh.com
hmac-sha256@ssh.com
hmac-sha-384@ssh.com
hmac-sha2-512
hmac-sha512@ssh.com
```

Multiple MACs can be specified as a comma-separated list. Special values for this option are:

- `Any` - includes all supported MACs including `none`

- `AnyStd` - includes MACs from the IETF SSH standards (`hmac-md5`, `hmac-md5-96`, `hmac-sha1`, `hmac-sha1-96`, `hmac-sha2-256`, `hmac-sha2-512`)

- `none` - no cryptographic data integrity method is used

- `AnyMac` - the same as `Any` but excludes `none`

- `AnyStdMac` - the same as `AnyStd` but excludes `none`

The default algorithms are `hmac-sha1`, `hmac-sha1-96`, `hmac-sha2-256`, `hmac-sha256-2@ssh.com`, `hmac-sha224@ssh.com`, `hmac-sha256@ssh.com`, `hmac-sha384@ssh.com`, `hmac-sha2-512` and `hmac-sha512@ssh.com`.

MaxBroadcastsPerSecond

Specifies how many UDP broadcasts the server handles per second. The default value is 0 and no broadcasts are handled at all. Broadcasts that exceed the limit are silently ignored. Received unrecognized UDP datagrams also consume the capacity defined by this option.

MaxConnections

Specifies the maximum number of connections that **sshd2** will handle simultaneously. This is useful against flooding attacks (attempts to interrupt the server from working properly by opening a high number of new connections). The argument is a positive number. The value `0` means that the number of connections is not limited. The default is 1000.

> ℹ️ **Note**
>
> `MaxConnections` must be greater than 1 when load control is used. For more information on load control, see Section 4.9.4.

Note that by using (at least) xinetd you achieve the same effect on a more generic level.

NoDelay

If set to `yes`, enables socket option `TCP_NODELAY`. The argument must be `yes` or no. The default is `no`.

PasswdPath

Specifies the location of the **passwd** program (or equivalent). By default this is set to where the `configure` script found it. This program will be run with the privileges of the user logging in.

PasswordGuesses

Specifies the number of tries that the user has when using password authentication. The default is 3.

PermitEmptyPasswords

When password authentication is allowed, this keyword specifies whether the server allows login to accounts with empty password strings. The argument must be `yes` or `no`. The default is `no`.

PermitRootLogin

Specifies whether root can log in using **sshg3**. The options are `yes`, `nopwd`, and `no`.

The default is `yes`, allowing root logins through any of the authentication types allowed for other users. The `nopwd` value disables password-authenticated root logins. The `no` value disables root logins through any of the authentication methods. (The values `nopwd` and `no` are equivalent unless you have some other means of authentication for root, e.g. public key.)

Root login with public-key authentication when the command option has been specified will be allowed regardless of the value of this setting (which may be useful for taking remote backups even if root login is normally not allowed).

PidFile

Specifies the file where the process ID of the server is written. The default is `/opt/tec-tia/var/run/sshd2_22.pid`.

Port

Specifies the port number that **sshd2** listens to. The current default is 22.

PrintMotd

Specifies whether **sshd2** should print `/etc/motd` (message of the day) when a user logs in interactively. The default is `yes`. The argument must be `yes` or `no`.

ProxyServer

With this option, **sshd2** can use a SOCKS (v4 or v5) or an HTTP proxy when a client forwards a connection. The server will use the value of this option when connecting. With SOCKS, you can specify whether to use SOCKS5 with the option `UseSocks5`.

The format of the variable is `socks://username@socks_server:port/network/netmask,network/net-mask ...` (with SOCKS proxy) or `http://username@socks_server:port/network/netmask,net-work/netmask ...` (with HTTP proxy).

For instance, by setting `ProxyServer` to

`socks://mylogin@socks.ssh.com:1080/203.123.0.0/16,198.74.23.0/24`

host `socks.ssh.com` and port `1080` are used as your SOCKS server for connections outside of networks `203.123.0.0` (16-bit domain) and `198.74.23.0` (8-bit domain). Those networks are connected directly.

If this option is set, it should almost always contain the local loopback network (127.0.0.0/8) as a network that is connected directly.

This option and the option `SocksServer` behave identically. Specifying both will cause the later definition to override the first.

PublicHostKeyFile

Specifies the file containing the public host key. The default is `/opt/tectia/etc/hostkey.pub`.

QuietMode

If set to `yes`, nothing is logged in the system log, except fatal errors. The argument must be `yes` or `no`. The default is `no`.

RandomSeedFile

Specifies the name of the random seed file.

RekeyIntervalSeconds

The number of seconds after which the key exchange is done again. The default is 3600 seconds (1 hour). The value 0 (zero) turns rekey requests off. This does not prevent the client from requesting rekeys. Other clients (not **sshg3**) may not have rekey capabilities implemented correctly, and they might not be able to handle rekey requests. This means that they may possibly close the connection or even crash.

RequiredAuthentications

Analogous to **AllowedAuthentications**, with one difference: the authentication methods listed here must all succeed before a user is considered authenticated. Leaving this list empty is equivalent to not using the option at all.

If this option is set, AllowedAuthentications is ignored.

> ℹ️ **Note**
>
> Versions of **sshd2** before 3.1.0 required RequiredAuthentications to be a subset of AllowedAuthentications. This is no longer the case.

RequireReverseMapping

This is used to check whether the host name DNS lookup must succeed when checking whether connections from hosts are allowed using **AllowHosts** and **DenyHosts**. If this is set to yes, and the name lookup fails, the connection is denied. If set to no, and name lookup fails, the remote host's IP address is used to check whether it is allowed to connect. This is probably not what you want if you have specified only host names (not IP addresses) with {Allow,Deny}Hosts. See also ResolveClientHostName. The argument must be yes or no. The default is no.

ResolveClientHostName

This parameter controls whether **sshd2** will try to resolve the client IP at all. This is useful when you know that the DNS cannot be reached, and the query would cause additional delay in logging in. Note that if you set this to no, you should not set RequireReverseMapping to yes. The argument must be yes or no. The default is yes.

SettableEnvironmentVars

This keyword can be followed by any number of patterns, separated by commas. Patterns are matched using the egrep syntax (see sshregex(1)), or the syntax specified in the metaconfiguration header of the configuration file. You can use the comma ',' character in the patterns by escaping it with a backslash '\'. By default, no environment variables can be set (but the default /opt/tectia/etc/sshd2_config file specifies some common and safe environment variables).

With this option, you can allow setting some or all environment variables. This option is used to check whether setting is allowed by the client (**sshg3**), by the user's `$HOME/.ssh2/environment` file or public key options. This is not used when setting variables from `/etc/environment` or other "root-only" files, as the user does not have control over those anyway.

Note that this option only changes the settings of environment variables before the user's shell is run. After that, the users are of course free to set whichever variables they want in the environment.

SftpDNSAllocWTO

If set to `yes`, output IGD101I and IGD104I messages on console upon allocation and deallocation of new SMS datasets. The default is `no`.

SftpSysLogFacility

As **SysLogFacility**, but defines the log facility the **sft-server-g3** subsystem will use. The default is `DAEMON`.

SftpSmfType

Defines the SMF record type used in SMF accounting in **sshd2** and the **sft-server-g3** subsystem. By default, this has no value, i.e. no accounting is performed. The possible values are: `none`, `TYPE119`.

ShellAccountCodeset

Specifies the code set of terminal session data in user's shell. The default is `IBM-1047`.

ShellAccountLineDelimiter

Specifies the line delimiter of terminal session data in user's shell. Special values for this option are `DOS`, `MAC`, `MVS` and `UNIX`. The default is `MVS`.

ShellConvert

Defines whether conversions are to be done for shell access and remote command execution. Possible values are `yes` and `no`. The default is `yes`.

ShellTransferCodeset

Defines the coded character set (CCS) on the line. The default is `ISO8859-1`.

ShellTransferLineDelimiter

Specifies the line delimiter of terminal session data during transfer. Special values for this option are `DOS`, `MAC`, `MVS` and `UNIX`. The default is `UNIX`.

ShellTranslateTable

Defines the MVS translate table to be used from line to shell.

SocksServer

Equal to **ProxyServer**.

StrictModes

Specifies whether **sshd2** should check file modes and ownership of the user's home directory and `.rhosts` files before accepting login using host-based authentication. This is normally desirable because novices sometimes accidentally leave their directory or files world-writable, in which case anyone can edit the `.rhosts` files.

If this is set, permissions are checked during public-key authentication for the user's `.ssh2` directory, public keys used and the `authorization` file.

Also the hostkey is checked for invalid permissions. The hostkey must only be readable and writable by the root user, the user public keys and `authorization` file must only be writable by the user. The permission check of the user's `.ssh2` directory (and with host-based authentication, the user's home directory) can be further controlled by using the `StrictModes.UserDirMaskBits` configuration option.

The argument must be `yes` or `no`. The default is `yes`.

StrictModes.UserDirMaskBits

Specifies the permission mask for the user's `.ssh2` directory if `StrictModes` configuration option is used. The bits set with this option are not allowed to be set in the actual permissions. This means that with `StrictModes` and this option set to `"077"`, the user's `.ssh2` directory may not have any permissions to group or others (only for the user). The default is `"022"`.

Subsystem-<subsystem name>

Specifies a subsystem. The argument is a command that is executed when the subsystem is requested.

sftpg3 uses a subsystem of **sshd2** to transfer files securely. In order to use the SFTP server, you must have the following subsystem definition:

```
subsystem-sftp sft-server-g3
```

SysLogFacility

Gives the facility code that is used when logging messages from sshd2. The possible values are: `DAEMON`, `USER`, `AUTH`, `LOCAL0`, `LOCAL1`, `LOCAL2`, `LOCAL3`, `LOCAL4`, `LOCAL5`, `LOCAL6`, `LOCAL7`. The default is `AUTH`.

TcpListenBacklog

TcpListenBacklog specifies the backlog value in listen(). The default value for TcpListenBacklog is `16`, which is currently hard coded. It allows the TCPIP stack to discard new connection request when the listen queue length is full.

TcpListenRate

TcpListenRate specifies the number of TCP Listen per second. If the TCP listen rate is hit, the listener will be paused for the amount of milliseconds defined with TcpListenPause. Valid value must be greater than zero. The default value is 5.

TcpListenPause

TcpListenPause specifies the number of milliseconds to pause. If the TcpListenRate is hit, the listener will be paused for the specified amount of milliseconds. Valid values are between 1 and 1000. The default value is 500 milliseconds (0.5 seconds).

Terminal.AllowUsers

Lists users that are allowed terminal access to the server host. This option can be followed by any number of patterns of the form user or user@host, separated by commas. The details explained under the **AllowHosts** option apply accordingly.

If this configuration option is used, only users that match the users listed under Terminal.AllowUsers may gain terminal access (provided that they are not restricted by other configuration options). By default, all users are allowed terminal access.

Note that all the other login authentication steps must still be successfully completed. Terminal.AllowUsers and Terminal.DenyUsers are additional restrictions.

Terminal.DenyUsers

Lists users that are denied terminal access to the server host. This is the opposite of Terminal.AllowUsers and works accordingly.

If a user matches a pattern in both Terminal.AllowUsers and Terminal.DenyUsers then terminal access is denied.

Note that when terminal access is denied so is remote command execution, forced commands (including commands related to public-key authentication and forced password changes), X11 forwarding, and agent forwarding. As a user has no shell access, any password changes (using system commands) will not be possible.

Access to subsystems (such as SFTP) and TCP tunneling (port forwarding) are still possible.

If a client requests terminal access (in addition to any other services, such as tunneling) the client may disconnect upon being refused terminal access. To prevent this, the client should be configured to not request terminal access, for example, by using the -S option in the **ssh** command (the option may vary with the **ssh** implementation).

Terminal.AllowGroups

Similar to `Terminal.AllowUsers` but matches groups rather than user names. The details explained under the **AllowGroups** option apply accordingly.

Terminal.DenyGroups

Similar to `Terminal.DenyUsers` but matches groups rather than user names. This is the opposite of `Terminal.AllowGroups` and works accordingly.

UseCryptoHardware

Specifies how cryptographic hardware is utilized. The value for this option is a comma-separated list of `algorithm:support_level` pairs. The list may start with a sole support level specifier.

Valid values for `support_level` are:

- `yes` - cryptographic hardware is used for this algorithm if available and software cryptography is used if hardware cryptography is not available

- `no` - software cryptography is used for this algorithm

- `must` - only cryptographic hardware is used for the algorithm

Valid values for `algorithm` are:

- `3des` - Triple DES symmetric cipher used for encrypting the session

- `aes` - AES symmetric cipher used for encrypting the session

- `sha` - SHA-1 or SHA-2 algorithm used for MAC (Message Authentication Code)

- `rng` - Random number generator. The hardware is used to generate entropy for the random seed at start-up and for reseeding the random number generator algorithm periodically during execution.)

> 🛈 **Note**
>
> If there is any use (as configured or by default) of an algorithm for which `support_level` is set to `must` and the relevant hardware support is not available, the server will refuse to start.

Example 1. Use cryptographic hardware for AES and SHA; all others should use software:

```
UseCryptoHardware        no,aes:must,sha:must
```

In this case, if any AES ciphers and/or SHA-based MACs are allowed in the configuration, and the relevant hardware support is not available, the server will refuse to start.

This could be coupled with allowing only AES ciphers and SHA-based MACs with the **Ciphers** and **MACs** options.

Example 2. Use cryptographic hardware for all algorithms except for AES:

```
UseCryptoHardware        yes,aes:no
```

UserConfigDirectory

Specifies where user-specific configuration data is found. With this the administrator can control those options that are usually controlled by the user. This is given as a pattern string which is expanded by **sshd2**.

%D is the user's home directory,

%U is the user's login name,

%IU is the user's user ID (uid),

%IG is the user's group ID (gid).

The default is %D/.ssh2.

UserKnownHosts

Specifies whether $HOME/.ssh2/knownhosts/ can be used to fetch host public keys when using host-based authentication. The argument must be yes or no. The default is yes.

UserSpecificConfig

As **HostSpecificConfig**, but these configuration files are read later, when the user name that the client is trying to log in as is known. Also the range of configuration options available is smaller, due to the fact that they would not make sense in these files. You can use patterns of form "user[%group][@host]", where user is matched with the user name and UID, group is matched with the user's primary and any secondary groups, both group name and GID, and host is matched as described under option **AllowHosts**.

See sshd2_subconfig(5) for more thorough documentation on what you can set in this subconfiguration file.

UseSocks5

Use SOCKS5 instead of SOCKS4 when connecting to remote host. Note that you have to set **SocksServer** to a meaningful value. The argument must be yes or no. The default is no (i.e. use SOCKS4).

VerboseMode

Causes **sshd2** to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems. Also causes **sshd2** to not fork on connection, so only one connection is handled at the time.

WTORoutingCodes

Specifies the z/OS routing codes for WTO messages. Valid values for routing codes are from 1 to 128. The default is 1,11.

ZcpuMathFacility

Controls the usage of the z13 Vector Facility. If set to `1` (default), **sshd2** will probe the availability of the z13 Vector Facility. If the facility is available, **sshd2** will use it. If `ZcpuMathFacility` is set to `0` or the facility is not available, **sshd2** will not use the z13 Vector Facility. The default is `1`.

sshd2_subconfig

sshd2_subconfig -- advanced configuration of sshd2 on z/OS

Description

You can also specify configuration options in so-called subconfiguration files, which have the same basic format as the main configuration file. The process forked to handle the user's connection reads these files. They are read at run-time, so if they are modified, it is not necessary to restart the server process.

If parsing of the subconfiguration files fails, the connection is terminated (for the host-specific subconfiguration) or access denied (for the user-specific subconfiguration) by the server.

Most of the configuration options that work in the main file work also in these, but some do not, where it either does not make sense to set them (e.g. ListenAddress and Port, which only affect the daemon process listening to the port, and would not affect that behavior in any way in a subconfiguration file) or it would be confusing (e.g. AllowUsers in user-specific subconfiguration, and AllowHosts in host-specific subconfiguration.).

The value for {Host,User}SpecificConfig keywords is a pattern-filename pair, separated by a whitespace. With **UserSpecificConfig**, the pattern is of format "user[%group][@host]", where the pattern user is matched with the user name and UID, group is matched with the user's primary and any secondary groups, both group name and GID, and host is matched as described under option **AllowHosts**. With **HostSpecific-Config**, the pattern is "host" (as in UserSpecificConfig).

Unlike sshd2_config, the subconfiguration files may have configuration blocks, or stanzas, in them. The subconfiguration heading is interpreted identically to what is described above, i.e. with UserSpecificConfig the pattern is of the format "user[%group][@host]", and with HostSpecificConfig the format is "host".

The subconfiguration files are divided into two categories: user-specific and host-specific. User-specific subconfiguration files are read when the client has stated the user name it is trying to log in with. At this point, the server will obtain additional information about the user: does the user exist, what is the user's UID, and what groups does the user belong to. With this information, the server can read the user-specific configuration files specified by UserSpecificConfig in the main **sshd2** configuration file.

The other category is host-specific configuration files, which are configured with the HostSpecificConfig variable. These files are read immediately after the daemon has forked a new process to handle the connection. Thus most configuration options can be set in these.

Note that it is possible to mix these configuration files. This is not recommended, because any global settings in these files would be set multiple times (which would not do any harm per se, but might lead to behavior not intended by the administrator).

Subconfigurations are really flexible, and because of that, dangerous if the logic of the files is not carefully planned. You can specify different authentication methods for different users, different banner messages for

people from certain hosts, and set log messages of certain groups to go to different files. There are a lot of possibilities here.

Options

Configuration variables that work everywhere, i.e. in the main file, the user-specific, and the host-specific configuration files:

AllowSHosts

AllowTcpForwarding

AllowedAuthentications

AuthHostbased.Cert.Required

AuthHostbased.Cert.ValidationMethods

AuthInteractiveFailureTimeout

AuthKbdInt.NumOptional

AuthKbdInt.Optional

AuthKbdInt.Plugin

AuthKbdInt.Required

AuthKbdInt.Retries

AuthPublicKey.Cert.Required

AuthPublicKey.Cert.ValidationMethods

AuthorizationEkInitStringMapper

AuthorizationEkInitStringMapperTimeout

AuthorizationEkProvider

AuthorizationFile

AuthPublicKey.MaxSize

AuthPublicKey.MinSize

CheckMail

DenySHosts

AllowAgentForwarding or **ForwardAgent**

HostbasedAuthForceClientHostnameDNSMatch

IdleTimeOut

IgnoreRhosts

IgnoreRootRhosts

KnownHostsEkProvider

PasswdPath

PasswordGuesses

PermitEmptyPasswords

PrintMotd

QuietMode

RekeyIntervalSeconds

RequiredAuthentications

SettableEnvironmentVars

`SftpSysLogFacility`

`ShellConvert`

`ShellAccountCodeset`

`ShellTransferCodeset`

`ShellTranslateTable`

`StrictModes`

`SysLogFacility`

`UserConfigDirectory`

`UserKnownHosts`

`VerboseMode`

Variables that work in the host-specific configuration file and the main file:

`AllowGroups`

`AllowTcpForwardingForGroups`

`AllowTcpForwardingForUsers`

`AllowUsers`

`BannerMessageFile`

`ChRootGroups`

`ChRootUsers`

`Ciphers`

`DenyGroups`

`DenyTcpForwardingForGroups`

`DenyTcpForwardingForUsers`

`DenyUsers`

`DisableVersionFallback`

`ExternalAuthorizationProgram`

`ForwardACL`

`IdentityDispatchUsers`

`KEXs`

`LoginGraceTime`

`MACs`

`PermitRootLogin`

sshregex

sshregex -- regular expressions used in file name globbing with scpg3, sftpg3 and configuration files

Description

This document describes the regular expressions (or globbing patterns) used in file name globbing with **scpg3** and **sftpg3** and in the sshd2_config and ssh-socks-proxy-config.xml configuration files. Regex syntax used with **scpg3** and **sftpg3** is zsh_fileglob.

Regex syntax: `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 the section called "Character Sets for egrep and zsh_fileglob".

$

   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.

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.

Regex syntax: `zsh_fileglob` (or `traditional`)

Patterns

The escape character is a backslash '\'. With this you can 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.

*

  Match any string consisting of zero or more characters. The characters can be any characters apart from slashes (/). However, the asterisk does not match a string if the string contains a dot (.) as its first character, or if the string contains a dot immediately after a slash. This means that the asterisk cannot be used to match file names that have a dot as their first character.

  If the previous character is a slash (/), or if an asterisk (*) is used to denote a match at the beginning of a string, it does match a dot (.).

  That is, the asterisk (*) functions as normal in Unix shell fileglobs.

?

  Match any single character except for a slash (/). However, do not match a dot (.) if located at the beginning of the string, or if the previous character is a slash (/).

  That is, the question mark (?) functions as normal in Unix shell fileglobs (at least in ZSH, although discarding the dot may not be a standard procedure).

**/

    Match any sequence of characters that is either empty, or ends in a slash. However, the substring '/.' is not allowed. This mimics the **/ construct in ZSH. (Please note that '**' is equivalent to '*'.)

E#

    Act as Kleene star, match E zero or more times.

E##

    Closure, match E one or more times.

(

    Start a capturing subexpression.

)

    End a capturing subexpression.

E|F

    Disjunction, match either E or F (inclusive). E is preferred if both match.

[

    Start a character set. (see below)

Character Sets for `egrep` and `zsh_fileglob`

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.

^ or ! (latter applies to ZSH_FILEGLOB)

    If immediately after the starting '[', denotes a complement: the whole character set will be complemented. Otherwise literal '^'.

[:alnum:]

    Characters for which 'isalnum' returns true (see ctype.h).

[:alpha:]

    Characters for which 'isalpha' returns true (see `ctype.h`).

[:cntrl:]

    Characters for which 'iscntrl' returns true (see `ctype.h`).

[:digit:]

    Characters for which 'isdigit' returns true (see `ctype.h`).

[:graph:]

   Characters for which 'isgraph' returns true (see `ctype.h`).

[:lower:]

   Characters for which 'islower' returns true (see `ctype.h`).

[:print:]

   Characters for which 'isprint' returns true (see `ctype.h`).

[:punct:]

   Characters for which 'ispunct' returns true (see `ctype.h`).

[:space:]

   Characters for which 'isspace' returns true (see `ctype.h`).

[:upper:]

   Characters for which 'isupper' returns true (see `ctype.h`).

[:xdigit:]

   Characters for which 'isxdigit' returns true (see `ctype.h`).

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.

Regex syntax: `ssh`

Patterns

The escape character is a tilde '~'. With this you can escape meta characters to use them in their plain character form.

ⓘ Note

   In configuration the backslash '\' is used to escape the list separator (',').

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.

{

    Start anonymous, non-capturing subexpression.

}

    End anonymous, non-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*?

    Act as Kleene star, but match non-greedily (lazy match).

E+

    Closure, match E one or more times.

E+?

    Closure, but match non-greedily (lazy match).

E?

    Option, match E optionally once.

E??

    Option, but match non-greedily (lazy match).

.

    Match ANY character, including possibly the NULL byte and the newline characters.

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.

E/n,/? , E/n,0/? , E/,n/? , E/0,n/? , E/n,m/?

    The lazy versions of above.

[

    Start a character set. See the section called "Character Sets for Regex Syntax `ssh`".

>C

One-character lookahead. 'C' must be either a literal character or parse as a character set. Match the empty string anywhere provided that the next character is 'C' or belongs to it.

<C

One-character lookback. As above, but examine the previous character instead of the next character.

$

Match the empty string at the end of the input.

^

Match the empty string at the start of the input.

Escaped Tokens for Regex Syntax **ssh**

~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.

~d

Match any digit, equivalent to [0:9].

~D

Match any character except a digit.

~s

Match a whitespace character (matches space, newline, line feed, carriage return, tab and vertical tab).

~S

Match a non-whitespace character.

~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.

Character Sets for Regex Syntax `ssh`

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:

:

A range operator, except immediately after '[', where it loses its special meaning.

- (minus sign)

Until the next +, the characters, ranges, and sets will be subtracted from the current set instead of adding. If appears as the first character after '[', start subtracting from a set containing all characters instead of the empty set.

+

Until the next -, the characters, ranges, and sets will be added to the current set. This is the default.

[:alnum:]

Characters for which 'isalnum' returns true (see `ctype.h`).

[:alpha:]

Characters for which 'isalpha' returns true (see `ctype.h`).

[:cntrl:]

Characters for which 'iscntrl' returns true (see `ctype.h`).

[:digit:]

Characters for which 'isdigit' returns true (see `ctype.h`).

[:graph:]

Characters for which 'isgraph' returns true (see `ctype.h`).

[:lower:]

Characters for which 'islower' returns true (see `ctype.h`).

[:print:]

Characters for which 'isprint' returns true (see `ctype.h`).

[:punct:]

Characters for which 'ispunct' returns true (see `ctype.h`).

[:space:]

Characters for which 'isspace' returns true (see `ctype.h`).

[:upper:]

Characters for which 'isupper' returns true (see `ctype.h`).

[:xdigit:]

Characters for which 'isxdigit' returns true (see `ctype.h`).

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.

Example: `[[:xdigit:]-a:e]` is typically equivalent to `[0123456789ABCDEFf]`.

Appendix B Default Configuration Files

The default server configuration files can be found in `/opt/tectia/etc`:

- `sshd2_config`: the main server configuration file

- `ssh_certd_config`: the Certificate Validator configuration file

For convenience, the contents of the default configuration files are also shown in Section B.1 and Section B.2.

B.1 Default `sshd2_config` Configuration File

The default `sshd2_config` configuration file is shown below. For descriptions of the configuration options, see sshd2_config(5)

```
## SSH CONFIGURATION FILE FORMAT VERSION 1.1
## REGEX-SYNTAX egrep
## end of metaconfig
## (leave above lines intact!)
##
## sshd2_config
##
## SSH Tectia Server 6.6 for IBM z/OS - SSHD2 Server Configuration File
##


## General

# Server Authentication: server keys in files
#       HostKeyFile                   hostkey
#       PublicHostKeyFile             hostkey.pub
#       HostCertificateFile           hostkey.crt # Comment out the pubkey
#                                                 # if cert is specified
# Server Authentication: server key and certificate in SAF
#       HostKeyEkProvider             "zos-saf"
#       HostKeyEkInitString           "KEYS(ID(SSHD2) RING(HOSTKEY) \
#LABEL('Host key label'))"
#       HostKey.Cert.Required         no
#
#       RandomSeedFile                random_seed
```

```
#        BannerMessageFile                 /opt/tectia/etc/ssh_banner_message
#        BannerMessageFile                 /etc/issue.net
#
#        VerboseMode                       no # For debugging only. See man page.
#        QuietMode                         no
#        SyslogFacility                    AUTH
#        SyslogFacility                    LOCAL7
#        SftpDSNAllocWTO                   NO
#        SftpExit                          " "
#        SftpSyslogFacility                DAEMON
#        SftpSmfType                       none
#        SftpSmfType                       TYPE119
#        SftpEndTransferWTO                NONE
#        SftpEndTransferWTO                SSZ3045I
#        SftpEndTransferWTO                "SSZ3045I Sftp transaction end:%n\
#%t File: %F%n\
#%t IP local/remote: %LI:%LP/%RI:%RP%n\
#%t Protocol version: %PV%n\
#%t Login method: %LM Cipher: %CS%n\
#%t Userid: %U Action: %A(%RC)%n\
#%t Bytes transferred: %TS"
#  where %F is the file/dataset name.
#        %LI and %LP are the local IP address and port number.
#        %RI and $RP are the remote IP address and port number.
#        %PV is the remote SSH protocol version.
#        %LM is the login mehtod (passowrd/publickey).
#        %CS is the cipher algorithm name.
#        %A  is the operation performed(UPLOAD/DOWNLOAD).
#        %RC is the operation status (SUCCEEDED/FAILURE).
#        %TS is the number of bytes transferred.
#        %t is eight space characters.
#        %n is newline character.
#        WTORoutingCodes           1,11
#        ZcpuMathFacility          1


## Communication with ssh-certd

#        CertdListenerPath                 /opt/tectia/var/run/ssh-certd-listener

## Network

#        Port                      22
#        AddressFamily             inet
#                                     inet:  IPv4 only
#                                     inet6: IPv6 only
#                                     any:   Ipv4 and IPv6
#
#        PidFile                   default
#        PidFile                   /opt/tectia/var/run/sshd2_22.pid
#        PidFile                   /opt/tectia/var/run/sshd2.pid
#        ListenAddress             any
```

```
#       ListenerRetryInterval          0
#       ListenerRetryInterval          60
#       ResolveClientHostName          yes
#       RequireReverseMapping          no
#       MaxBroadcastsPerSecond         0
#       MaxBroadcastsPerSecond         1
#       NoDelay                        no
#       KeepAlive                      yes
#       TcpListenBacklog               16
#       TcpListenRate                  5
#       TcpListenPause                 500


## Load Control

#       MaxConnections                 1000
#       LoadControl.Active             no
#       LoadControl.DiscardLimit       see below
#       LoadControl.WhitelistSize      1000
#
# MaxConnections is the limit of the number of concurrent connections. It must
# be greater than 1 when Load Control is active. The value 0 means that the
# number of connections is unlimited and causes Load Control to be disabled.
#
# Set LoadControl.Active to yes to enable load control.
#
# LoadControl.DiscardLimit must not be larger than MaxConnections. The default
# value is 90 % of MaxConnections.
#
# LoadControl.WhitelistSize is the number of distinct IP addresses that the
# whitelist can hold.


## Crypto

#       Ciphers                        aes128-ctr,aes192-ctr,aes256-ctr,\
#aes128-gcm@openssh.com,aes256-gcm@openssh.com,\
#aes128-cbc,aes192-cbc,aes256-cbc,3des-cbc
# Specifies the accepted encryption algorithms for connection security. It is a
# list of cipher names or one of the names Any, AnyCipher, AnyStd or
# AnyStdCipher.
# Any and AnyCipher include all the ciphers supported by Tectia.
# AnyStd and AnyStdCipher include ciphers listed in the SSH standards.
# Any and AnyStd also include "none", which means no encryption.
#

#       MACs                           hmac-sha1,hmac-sha1-96,hmac-sha2-256,\
#hmac-sha256-2@ssh.com,hmac-sha224@ssh.com,hmac-sha256@ssh.com,\
#hmac-sha384@ssh.com,hmac-sha2-512,hmac-sha512@ssh.com
# Specifies the accepted Message Authentication Codes for connection security.
# It is a list of MAC names or one of the names Any, AnyMAC, AnyStd or
# AnyStdMAC.
# Any and AnyMAC include all the MACs supported by Tectia.
```

```
# AnyStd and AnyStdMAC include the MACs listed in the SSH standards.
# Any and AnyStd also include "none", which means no message authentication.
#

#       Compressions                    none,zlib
# Specifies the accepted compression methods for connection.

#       KEXs                            ecdh-sha2-nistp521,\
#ecdh-sha2-nistp384,ecdh-sha2-nistp256,\
#diffie-hellman-group-exchange-sha256,\
#diffie-hellman-group14-sha256,\
#diffie-hellman-group14-sha256@ssh.com,\
#diffie-hellman-group16-sha512,diffie-hellman-group18-sha512,\
#diffie-hellman-group14-sha1
# A list of key exchange names or Any, AnyKEX, AnyStd or AnyStdKEX.
#

#       HostKeyAlgorithms               x509v3-ecdsa-sha2-nistp521,\
#x509v3-ecdsa-sha2-nistp384,x509v3-ecdsa-sha2-nistp256,\
#ecdsa-sha2-nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256,\
#x509v3-sign-dss,x509v3-sign-rsa,\
#rsa-sha2-256,rsa-sha2-512,ssh-dss,ssh-rsa,\
#x509v3-sign-dss-sha256@ssh.com,x509v3-sign-rsa-sha256@ssh.com,\
#ssh-dss-sha256@ssh.com,ssh-rsa-sha256@ssh.com
# A list of host key algorithm names or Any, AnyHostKeyAlgorithm, AnyStd or\
# AnyStdHostKeyAlgorithm.
#

#       RekeyIntervalSeconds        3600

## Crypto Hardware

#       UseCryptoHardware               yes
# Specifies whether hardware support is wanted for certain
# algorithms. The support levels are
#   no          do not use crypto hardware
#   yes         use crypto hardware if available
#   must        use crypto hardware, fail if not available
#
# The level may be given alone as a default for all algorithms or
# together with an algorithm. The algorithm names that may
# be used are:
#   rng         random number generator
#   sha         SHA1 and SHA2 digest algorithms (sha1 is equivalent)
#   aes         AES algorithms
#   3des        Triple DES
#
# UseCryptoHardware is a comma-delimited list of algorithm:support level
# pairs. It may start with a sole support level
#
# E.g. To use all available hardware support and fail if support for 3DES
```

```
#      or SHA is not available, specify "yes,aes:must,sha:must"
#
# On most IBM mainframe systems the following algorithms have hardware support:
# the ciphers "aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc" and the MACs
# "hmac-sha1,hmac-sha224@ssh.com,hmac-sha256@ssh.com,hmac-sha2-256,
# hmac-sha256-2@ssh.com, hmac-sha384@ssh.com,hmac-sha2-512,
# hmac-sha512@ssh.com". The support is provided by
# the CPACF or IBM cryptographic card facility together with ICSF.
#
#      CryptoCardCipherIOThreshold     65536
# Specifies the minimum size of cipher request that will be routed to
# IBM cryptographic co-processor card (CEX), if the card is available and
# UseCryptoHardware is set to yes/must, for cipher processing.  If the
# request size is less than the CryptoCardCipherIOThreshold value, the
# cipher request will be routed to CPACF facility.  Special values are
#   0                route all cipher requests to IBM cryptographic
#                    co-processor card
#   65536 or higher  route all cipher requests to CPACF facility
#
#      CryptoCardMACGenerate           no
# Specifies whether to route MAC generation request to IBM cryptographic
# co-processor card (CEX).  If it is set to yes, MAC generation request will
# route to IBM cryptographic co-processor card (CEX), if the card is available
# and UseCryptoHardware is set to yes/must, for MAC generation processing.

## User

#      PrintMotd                    yes
#      CheckMail                    yes
#      StrictModes                  yes
# Specifies 1 hour (you can also use 'w' for week, 'd' for day, 'm' for
#                  minute, 's' for seconds)
#      IdleTimeOut                  1h
# without specifier, the default number is in seconds
#      IdleTimeOut                  3600
#
#      UserConfigDirectory          "%D/.ssh2"
#      UserConfigDirectory          "/opt/tectia/etc/auth/%U"
#      AuthorizationFile            authorization
#
# Authorized keys file directive can be used in enabling public-key
# authentication against legacy authorized_keys file that contains
# several keys in single file.
#      AuthorizedKeysFile           "authorized_keys"
#      AuthorizedKeysFile           "%D/.ssh/authorized_keys"
#
# This variable is set here, because by default it is empty, and so no
# variables can be set. Because of that, we set a few common ones here.
       SettableEnvironmentVars      LANG,\
LC_(ALL|COLLATE|CTYPE|MONETARY|NUMERIC|TIME),PATH,TERM,TZ,SSH.*
```

```
## Conversion on terminal session


#       ShellTransferCodeset          ISO8859-1
#       ShellTransferLineDelimiter    UNIX
#       ShellAccountCodeset           IBM-1047
#       ShellAccountLineDelimiter     MVS
#       ShellTranslateTable           ""
#       ShellConvert                   yes


## Tunneling


#       AllowTcpForwarding            yes
#       AllowTcpForwardingForUsers    sjl, ra-user@remote\.example
#       DenyTcpForwardingForUsers     2[[:digit:]]*4,peelo
#       AllowTcpForwardingForGroups   privileged_tcp_forwarders
#       DenyTcpForwardingForGroups    coming_from_outside
#
#       AllowLocalForwarding          no
        AllowLocalForwarding          yes


# Local port forwardings to host 10.1.0.25 ports 143 and 25 are
# allowed for all users in group users.
# Note that forwardings using the name of this host will be allowed (if
# it can be resolved from the DNS).
#
#       ForwardACL allow local .*%users \i10\.1\.0\.25%(143|25)
#
# Local port forwardings requested exactly to host proxy.example.com
# port 8080 are allowed for users that have 's' as first character
# and belong to the group with group ID (GID) 10:
#
#       ForwardACL allow local s.*%10 proxy\.example\.com%8080
#
# Remote port forwarding is denied for all users to all hosts:
#       ForwardACL deny remote .* .*



## Authentication


## publickey and password allowed by default
#       AllowedAuthentications        publickey,password
#       AllowedAuthentications        hostbased,publickey,password
#       AllowedAuthentications        hostbased,publickey,keyboard-interactive
#       RequiredAuthentications       publickey,password
#       LoginGraceTime                600
#       AuthInteractiveFailureTimeout 2
#
#       HostbasedAuthForceClientHostnameDNSMatch no
#       UserKnownHosts                yes
#
#       AuthPublicKey.MaxSize         0
```

```
#       AuthPublicKey.MinSize           0
#       AuthPublicKey.Algorithms        AnyStdPublicKeyAlgorithm
#
#       AllowAgentForwarding            yes

#       AuthKbdInt.NumOptional          0
#       AuthKbdInt.Optional             password,plugin
#       AuthKbdInt.Required             password
#       AuthKbdInt.Retries              3
#
#       PermitEmptyPasswords            no
#       PasswordGuesses                 3
#
## publickey authentication with certificates in SAF
# Users logging in with name "-" need SAF certificate
#       IdentityDispatchUsers                   -
#
# All users logging in need SAF certificate
#       IdentityDispatchUsers                   .*
#
#       AuthPublicKey.Cert.ValidationMethods    saf
#
# Certificate is also validated in ssh-certd
#       AuthPublicKey.Cert.ValidationMethods    saf,tectia
#
# Client must send user certificate
#       AuthPublicKey.Cert.Required             no
#
#       AuthorizationEkProvider         "zos-saf:KEYS(ID(%UU) RING(%UU))"
#       AuthorizationEkProvider         "zos-saf:[USERNAME=%U UID=%IU GID=%IG]"
#       AuthorizationEkInitStringMapper /home/SSHD2/mapper.sh
#       AuthorizationEkInitStringMapperTimeout 0   # 0 = Timeout disabled
#
## hostbased authentication with certificates in SAF
#       AuthHostbased.Cert.ValidationMethods    saf
#
# Certificate is also validated in ssh-certd
#       AuthHostbased.Cert.ValidationMethods    saf,tectia
#
# Client must send host certificate
#       AuthHostbased.Cert.Required  no
#       KnownhostsEkProvider        "zos-saf:KEYS(ID(SSHD2) RING(KNOWNHOSTS))"
#


# To enable authentication time password changing (instead of the old
# forced command style), uncomment the following line:


#       AuthPassword.ChangePlugin       ssh-passwd-plugin


# (this will also be used by the "password" submethod in
#  keyboard-interactive).
```

```
## Host restrictions


#       AllowHosts                  localhost, example\.com, friendly\.example
#
## Next one matches with, for example, taulu.foobar.com, tuoli.com, but
## not tuoli1.com. Note that you have to input string "\." when you want it
## to match only a literal dot. You also have to escape "," when you
## want to use it in the pattern, because otherwise it is considered a list
## separator.
##
##      AllowHosts               t..l.\..*
##
## The following matches any numerical IP address (yes, it is cumbersome)
##
##      AllowHosts               ([[:digit:]]{1\,3}\.){3}[[:digit:]]{1\,3}
##
## Same thing is achieved with the special prefix "\i" in a pattern.
## This means that the pattern is only used to match IP addresses.
##
## Using the above example:
##
##      AllowHosts               \i.*
##
## You can probably see the difference between the two.
##
## Also, you can use subnet masks, by using prefix "\m"
##
##      AllowHosts               \m127.0/8
## and
##      AllowHosts               \m127.0.0.0/24
##
## would match localhost ("127.0.0.1").
##
#       DenyHosts                   evil\.example, aol\.example
#       AllowSHosts                 trusted\.host\.example
#       DenySHosts                  not\.quite\.trusted\.example
#       IgnoreRhosts                no
#       IgnoreRootRHosts            no
# (the above, if not set, is defaulted to the value of IgnoreRHosts)


## User restrictions
# User and group names must be in uppercase.

#       AllowUsers                  SJ.*,S[[:digit:]]*,S(JL|AMZA)
#       DenyUsers                   SKUUPPA,WAREZDUDE,31373
#       DenyUsers                   DON@example\.org
#       AllowGroups                 STAFF,USERS
#       DenyGroups                  GUEST,ANONYMOUS
#       PermitRootLogin             yes
#       PermitRootLogin             nopwd
```

```
## Chrooted environment
# User and group names must be in uppercase.

#       ChRootUsers                     ANONYMOUS,FTP,GUEST
#       ChRootGroups                    SFTP,GUEST


## Subsystem definitions

# Subsystems do not have defaults, so this is needed here (uncommented).
#       subsystem-sftp                  sftp-server
        subsystem-sftp                  /opt/tectia/libexec/sft-server-g3
# Also internal SFTP subsystem can be used.
#       subsystem-sftp                  internal://sftp-server


## Subconfiguration
# There are no default subconfiguration files. When specified the last
# obtained keyword value will prevail. Note that the host-specific files
# are read before the user-specific files.
# User and group names must be in uppercase.

# Following matches (from) any host:
#
#       HostSpecificConfig .* /opt/tectia/etc/subconfig/host_ext.example
#
# Following matches to subnet mask:
#
#       HostSpecificConfig \m192.168.0.0/16 \
#/opt/tectia/etc/subconfig/host_int.example
#
# Following matches to users from ssh.com that have two character
# username or username is SJL and belong to group WHEEL or WHEEL[0-9]:
#
#       UserSpecificConfig (..|SJL)%WHEEL[[:digit:]]?@ssh\.com \
#/opt/tectia/etc/subconfig/user.example
#
# Following matches to the user ANONYMOUS from any host:
#
#       UserSpecificConfig ANONYMOUS@.* \
#/opt/tectia/etc/subconfig/anonymous.example
```

B.2 Default `ssh_certd_config` Configuration File

The default ssh_certd_config configuration file is shown below. For descriptions of the configuration options, see ssh_certd_config(5)

```
## SSH CONFIGURATION FILE FORMAT VERSION 1.1
## REGEX-SYNTAX egrep
## end of metaconfig
## (leave above lines intact!)
```

```
## ssh_certd_config
## &fullname; - Certificate Validator Configuration File ##


UseSSHD2ConfigFile                      sshd2_config


## General


#       VerboseMode             no
#       QuietMode               no
#       SyslogFacility          AUTH
#       RandomSeedFile          /opt/tectia/etc/random_seed


## Certificate configuration


#       CertCacheFile           /var/spool/ssh-certd-cache
#       SocksServer             socks://mylogin@socks.example.com:1080
#       UseSocks5               no
#       OCSPResponderURL        http://example.com:8090/ocsp-1/
#       LdapServers             ldap://example.com:389


## X.509 certificate of the root CA which is trusted when validating
#  user certificates.


#       Pki                     ca-certificate,use_expired_crls=3600
#       PkiDisableCrls          no
#       Mapfile                 ca-certificate.mapfile


## External key provider for fetching root CA X.509 certificates
#  from RACF or equivalent. The certificates found from the specified
#  ring(s)/label(s) are trusted when validating user certificates.


#       PkiEkProvider           "zos-saf:KEYS(ID(SSHD2) RING(SSH-PKI))"
#       PkiDisableCrls          no
#       Mapfile                 ca-certificate.mapfile


## External key provider for fetching root CA X.509 certificates
#  from RACF or equivalent. The certificates found from the specified
#  ring(s)/label(s) are trusted when validating remote host certificates
#  in hostbased user authentications.


#       HostCAEkProvider        "zos-saf:KEYS(ID(SSHD2) RING(SSH-HOSTCA))"


## CRL autoupdate


#       CrlAutoUpdate           yes,update_before=30,min_interval=30


## CRL manual update


#       CrlPrefetch             3600 ldap://example.com/
```

Appendix C IPv6 Support on Tectia Server and client tools for IBM z/OS

Tectia Server and client tools for IBM z/OS version 6.3 and later support IPv6. The products should function correctly in properly configured:

* IPv4-only networks

* IPv6-only networks

* Mixed IPv4 and IPv6 networks.

This has impact on several areas where hosts or addresses may be defined.

The purpose of this appendix is to provide an overview of the configuration of Tectia Server and client tools for IBM z/OS to use IPv6. We first discuss the server (in Section C.1), then the two areas relevant to client operations, the Connection Broker (in Section C.2.1) and the client itself (in Section C.2.2). Finally, some examples of tunneling involving IPv6 are discussed (in Section C.2.3).

C.1 Server Configuration and Use

In terms of how IPv4 and IPv6 addresses are supported, the server, **sshd2**, is managed by the following keywords in the `sshd2_config` configuration file:

`AddressFamily { inet | inet6 | any }`
> This keyword specifies which address families may be used. It may be supplied exactly once or be omitted.
>
> Valid values for `AddressFamily` are:
>
> * `inet` (*default*): Accepts IPv4 addresses only. IPv6 addresses are ignored.
>
> * `inet6`: Accepts IPv6 addresses only.
>
> * `any`: Accepts both IPv4 and IPv6 addresses.

The command-line options `-4` and `-6` correspond to `inet` and `inet6`. If the command-line options are specified, they override the configuration keyword.

If no configuration keyword or command-line option is supplied, the default `inet` is used.

ListenAddress { *IPv4 address* | *IPv6 address* } [*port*]

This keyword specifies the network interface(s) on which the server will listen. It may be supplied multiple times, or be omitted:

- If `ListenAddress` is omitted, **sshd2** listens on the IPv4 `0.0.0.0` address or the IPv6 `::` address, or both, depending on the address family selected as above.

- Multiple listening sockets are opened in the order given by the sequence of `ListenAddress` keywords found.

Listen addresses must accord with the `AddressFamily` specification: an address ruled out by the address family selected will be ignored.

Optionally, the port on which to listen may be specified here, thus overriding any `Port` configurations and the default port 22.

The command-line option `-o ListenAddress='`*ip-address* `[`*port*`]`' may also be used to specify interfaces on which the server is to listen. These are additional to any interfaces specified using configuration keywords. Each `-o ListenAddress` option must be followed by one address-port definition, consisting of an IPv4 or IPv6 address followed by a space, and an optional port number. If the port is missing, the last `Port` configuration keyword or the default port 22 is used.

Note that the `-p` port number option will override any port number specified via the configuration keyword `Port` or `ListenAddress`, but it does not affect any port numbers supplied with the `-o ListenAddress` option.

An IPv6 address may be either a link-local address or a global address; the server will listen all scopes of a link-local address if one is specified.

The following table demonstrates some of the allowed and invalid combinations of the above-mentioned configuration keywords and command-line options, and their resultant effect:

```
Address   -4 / -6  Listen       -o             Listen on
Family             Address       ListenAddress  interface(s), port(s)

*         -4       none         127.0.0.1      127.0.0.1
*         -4       none         127.0.0.1 23   127.0.0.1:23
*         -4       none         ::1            error
*         -4       none         any            0.0.0.0:22
*         -4       none         any 23         0.0.0.0:23
*         -4       none         none           0.0.0.0:22
*         -6       none         0.0.0.0        error
*         -6       none         ::1            [::1]:22
```

*	-6	none	127.0.0.1	error
*	-6	none	::	[::]:22
*	-6	none	any	[::]:22
*	-6	none	none	[::]:22
any	-6	0.0.0.0	none	0.0.0.0:22
any	none	::1	127.0.0.1 23	[::1]:22, 127.0.0.1:23
any	none	::1 23	none	[::1]:23
any	none	127.0.0.1	10.1.1.1 23	127.0.0.1:22, 10.1.1.1:23
any	none	127.0.0.1	::1	127.0.0.1:22, [::1]:22
any	none	127.0.0.1 23	none	127.0.0.1:23
any	none	127.0.0.1	fe80::100:10 23	127.0.0.1:22, [fe80::100:10]:23
any	none	127.0.0.1	none	127.0.0.1:22
any	none	::1	none	[::1]:22 ❶
any	none	[::]	none	0.0.0.0:22, [::]:22
any	none	none	::	0.0.0.0:22, [::]:22
any	none	none	::1	[::1]:22
any	none	none	127.0.0.1	127.0.0.1:22
any	none	none	127.0.0.1 23	127.0.0.1:23
any	none	none	any	0.0.0.0:22, [::]:22
any	none	none	any 23	0.0.0.0:23, [::]:23
any	none	none	none	0.0.0.0:22, [::]:22
inet6	-4	::1	none	[::1]:22
inet6	-6	0.0.0.0	none	error
inet6	none	::1	:: 23	[::1]:22, [::]:23
inet6	none	::1 23	::	[::1]:23, [::]:22
inet6	none	::1 23	127.0.0.1	[::1]:23
inet6	none	::1 23	fe80::100:10	[::1]:23, [fe80::100:10]:22
inet6	none	::1 23	none	[::1]:23
inet6	none	127.0.0.1	::	[::]:22
inet6	none	127.0.0.1	none	error
inet6	none	::1	none	[::1]:22 ❷
inet6	none	none	127.0.0.1	error ❸
inet6	none	none	:: 23	[::]:23
inet6	none	none	fe80::100:10 23	[fe80::100:10]:23
inet6	none	none	none	[::]:22
inet	-6	0.0.0.0	none	0.0.0.0:22
inet	none	::1	127.0.0.1	127.0.0.1:22
inet	none	127.0.0.1	10.1.1.1 23	127.0.0.1:22, 10.1.1.1:23
inet	none	127.0.0.1	::	127.0.0.1:22
inet	none	127.0.0.1 23	10.1.1.1 22	127.0.0.1:23, 10.1.1.1:22
inet	none	127.0.0.1	:: 23	127.0.0.1:22
inet	none	127.0.0.1 23	none	127.0.0.1:23
inet	none	127.0.0.1	none	127.0.0.1:22
inet	none	::1	none	error
inet	none	none	::	error
inet	none	none	none	0.0.0.0:22
none	-4	::1	127.0.0.1	127.0.0.1:22
none	-4	::1	none	error
none	-6	0.0.0.0	none	error
none	none	none	[::] 23	error
none	none	none	any	0.0.0.0:22

none	none	none	any 23	0.0.0.0:23
none	none	none	::	error
none	none	none	none	0.0.0.0:22

❶❷　Subsequent **sshg3 ::1#22** succeeds; **sshg3 127.0.0.1#22** fails.

❸　`AddressFamily` is overridden if explicit address is given.

Running the server with the verbose option may help to clarify what interfaces are being listened on. See sshd2(8) and sshd2_config(5) for more details, as well as for where hosts may be specified or matched by IPv6 address (e.g. `sshd2_config` keywords **AllowHosts**, **AllowSHosts**, **ProxyServer**, etc.).

C.2 Client Configuration and Use

C.2.1 Connection Broker

With respect to IPv6, the client is affected by the Connection Broker configuration as given in the following files:

- Default configuration file (optional): `/opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-config-default.xml`

- Global configuration file (optional): `/opt/tectia/etc/ssh-broker-config.xml`

- User-specific configuration file (optional): `$HOME/.ssh2/ssh-broker-config.xml`

- A configuration file explicitly specified by the `-f` command-line option.

The common DTD for all these XML configuration files is: `/opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd`. You can see the contents of the DTD also in the *Tectia Server for IBM z/OS User Manual*.

All of the above-mentioned configuration files are optional; if none are provided, the defaults, as defined in the DTD, are used. Files later in the above list override settings in their predecessors, if any.

For the purposes of IPv6 support, the `address-family` sub-element of the `default-settings` element is the first consideration:

```
<!-- Both ipv4 and ipv6 are enabled by default -->
<!ENTITY default-address-family-type          "any">

<!-- address-family mode setting ipv4 & ipv6-->
<!ELEMENT address-family      EMPTY>
<!ATTLIST address-family
         type          (any|inet|inet6) "&default-address-family-type;">
```

These lines in the DTD mean that `address-family` may be coded as `any`, `inet`, or `inet6` and that if not specified, it defaults to `any`. Note that the default on the client side is the more permissive setting to support both IPv4 and IPv6, whereas the server defaults are generally more conservative, supporting IPv4 only, in the absence of explicit instructions.

In many other places in the Connection Broker configuration, such as in tunnel definitions, IP addresses or host names may be specified. Depending on the `address-family` chosen, IPv4 or IPv6 addresses are allowable.

C.2.2 Client

Client refers to **sshg3**, **scpg3**, **sftpg3**, and several utilities such as **ssh-keyfetch-g3** and **ssh-keydist-g3.** Their behavior in terms of IP `address-family` may be modified by:

- The Connection Broker configuration through which the clients communicate with servers (discussed above)

- The use of a connection profile defined in the Connection Broker configuration

- Command-line options

The host attribute of a connection profile may specify either an IP address or a host name. Whether IPv4 or IPv6 is allowed and whether lookups of host names use IPv4 or IPv6 addresses is controlled by the address family selected for the connection, based on the evaluation of the Connection Broker configurations and broker and client command-line options. In fact, this is the general rule whenever client configurations or operands refer to an address or host, so with that in mind it should be possible to deduce the effect of any such reference in terms of address family.

Command-line options for the client that affect the use of IPv4 or IPv6 are, in the first instance, the `-4` and `-6` options. These affect the process of DNS lookup, having a restrictive effect on the `getaddrinfo()` library function used to resolve hosts. In the absence of either option, the default `unspec` address family is used, meaning that address resolution may return both IPv4 and IPv6 addresses. The `-4` option results in only IPv4 addresses being returned, while `-6` results in only IPv6 addresses. Note that these options control just the action of `getaddrinfo()`: they determine how names and addresses are resolved for the purpose of the client making connections; they do not determine the address family in use, which is instead controlled at the Connection Broker level.

Addresses or hosts may be used as operands by clients as connection targets and tunnel listen interfaces and targets. Explicit IPv4 or IPv6 addresses may be coded, depending on the configuration and options in effect, and host names are likewise resolved to IPv4 or IPv6 addresses or both. Clients may connect to a server listening on a link-local IPv6 address, but only if the correct `scope_id` is supplied, e.g., `fe80::100:00%1`, etc.

The following table demonstrates some of the possible combinations and their effects:

Address Family	-4 / -6 option	Address operand	Connect to

```
*            *            none               error
none         none         localhost          first resolved usable address
none         -4           localhost          127.0.0.1
none         -6           localhost          ::1 (fails unless resolves as IPv6)
none         none         127.0.0.1          127.0.0.1
none         -4           127.0.0.1          127.0.0.1
none         -6           127.0.0.1          127.0.0.1
none         none         ::1                ::1
none         -4           ::1                ::1
none         -6           ::1                ::1
none         *            fe80::100:00       error (link local requires scope)
none         *            fe80::100:00%1     fe80::100:00%1
none         *            fd02:200:bb::4     fd02:200:bb::4 (global scope address)
'any'                                        see AddressFamily none
'inet'       none         localhost          first resolved usable address
'inet'       -4           localhost          127.0.0.1
'inet'       -6           localhost          ::1 (fails unless resolves as IPv6)
'inet'       *            ::1                ::1
'inet'       *            fe80::100:00       error (link local requires scope)
'inet'       *            fe80::100:00%1     fe80::100:00%1
'inet6'      none         localhost          ::1 (fails unless resolves as IPv6)
'inet6'      -4           localhost          127.0.0.1
'inet6'      -6           localhost          ::1 (fails unless resolves as IPv6)
'inet6'      *            127.0.0.1          127.0.0.1
'inet6'      *            ::1                ::1
'inet6'      *            10.0.0.1           10.0.0.1
'inet6'      *            fe80::100:00       error (link local requires scope)
'inet6'      *            fe80::100:00%1     fe80::100:00%1
*            *            fd02:200:bb::4     fd02:200:bb::4 (global scope address)
```

For the sake of brevity, this table uses the terms localhost, `127.0.0.1`, and `::1`, but of course localhost might be configured as `127.0.0.2`, `::2`, etc. The idea is that other types of address may be extrapolated from those given here.

C.2.3 Tunneling

Tunneling involves several addresses, as in, for example:

```
sshg3 -L 2001:[::1]:2002 127.0.0.1
```

meaning "connect to the default SSH port on `127.0.0.1` and start a tunnel on `localhost:2001` which will forward to `[::1]:2002`", or:

```
sshg3 -R 1234:remotehost:22 username@sshserver
```

meaning "connect to `sshserver` and start a tunnel listening on `localhost:1234` forwarding to `remote-host:22`", where `localhost` refers to the machine on which the tunnel is being created. In this case, the `AddressFamily` in effect determines whether IPv4 or IPv6 is used.

Defining the `listen` part of a tunnel follows the same rules with respect to IP addressing as does listening on the server. The connection end of the tunnel is defined according to the same rules as any client connection.

There is nothing new to take into account for IPv6 and tunnels apart from the points mentioned for servers and clients. Tunnels can be established in as complex a way as desired and IPv4 and IPv6 can be mixed as wished.

The following examples illustrate some tunnel commands and their effects. In the examples, `local1` and `remote1` have both IPv4 and IPv6 connectivity; `remote2` has IPv4 only:

```
local1                 172.29.127.54         fd02:2382:bb5a::54

remote1                172.29.127.36         fd02:2382:bb5a::36

remote2                172.29.127.227
```

Example 1: Local tunnel with IPv4 and IPv6

```
local1$ sshg3 -6 -L [::1]:4444:remote2:333 remote1#22422 ❶

local1$ netstat -tln
tcp        0        0 ::1:4444                  :::*                     LISTEN

local1$ sshg3 localhost#4444 ❷

remote2$ netstat -a ❸
SSHD2    00000010 0.0.0.0..333             0.0.0.0..0              Listen
SSHD2    000005FA 172.29.127.227..333    172.29.127.36..1160    Establsh

remote1$ netstat -a ❹
SSHD22   00000382 Establsh
  Local Socket:    172.29.127.36..1160
  Foreign Socket:  172.29.127.227..333
SSHD25   0000030E Listen
  Local Socket:    ::..22422
  Foreign Socket:  ::..0
SSHD26   0000037D Establsh
  Local Socket:    fd02:2382:bb5a::36..22422
  Foreign Socket:  fd02:2382:bb5a::54..44268
```

❶   Connect to port 22422 on `remote1`, preferring to resolve the host name to an IPv6 address, and opening a tunnel listening on port 4444 on this host's local `::1` interface, and which connects to port 333 on host `remote2`.

❷   Still on `local1`, connect to localhost, port 4444.

❸   This will take us to `remote2`, where the **netstat** command shows a connection from `remote1` to port 333.

❹   On `remote1`, the **netstat** command shows the incoming connection from `local1` to the server on port 22422 over IPv6, and the outgoing connection over IPv4 to port 333 on `remote2`.

Example 2: Remote tunnel with IPv4 and IPv6

```
local1$ sshg3 -R [fd02:2382:bb5a::36]:3333:remote2:333 remote1#22422 ❶


remote1$ netstat -c SERVER ❷
SSHD21   00000375 Listen
  Local Socket:    fd02:2382:bb5a::36..3333


remote1$ sshg3 -p3333 fd02:2382:bb5a::36 ❸


remote2$ netstat -a ❹
SSHD2    000005F5 172.29.127.227..333    172.29.127.54..39631    Establsh
```

❶   Connect to `remote1` via port 22422 and open a tunnel on the interface with address `fd02:2382:bb5a::36` and port 3333.

❷   The **netstat** command on `remote1` shows this listening socket. The other end of the tunnel goes to port 333 on `remote2`.

❸   A client on `remote1` connects to `[fd02:2382:bb5a::36]:3333` and ends up on `remote2`.

❹   The **netstat** command on `remote2` shows an established IPv4 connection between `remote2` and `local1`.

Appendix D Running the Server and SOCKS Proxy on Multiple TCP/IP Stack z/OS

Multiple instances of Tectia Server for IBM z/OS and Tectia SOCKS Proxy can be run on different TCP/IP stacks. In the following sections we give instructions for running two Servers or SOCKS Proxies on a dual stack z/OS machine, but the described setup can be extended to accommodate environments with more than two TCP/IP stacks.

D.1 Running Two Servers on a Dual TCP/IP Stack

In this example we have two started tasks for two **sshd2** servers: SSHD2 and SSHD2B.

1. **Configuration Files**

   Create two separate configuration files for the two servers to maintain their PID files distinct. In this example we have files `/opt/tectia/etc/sshd2_config` and `/opt/tectia/etc/sshd2b_config`.

2. **PID Files**

   The only difference between the two configuration files is the value of the **PidFile** keyword. It specifies the file where the process ID of the Server is written. For example, for SSHD2 in `/opt/tectia/etc/sshd2_config` the PidFile keyword has the default value:

   ```
   PidFile      /opt/tectia/var/run/sshd2_22.pid
   ```

   For SSHD2B in `/opt/tectia/etc/sshd2b_config` we specify the PID file sshd2b_22.pid:

   ```
   PidFile      /opt/tectia/var/run/sshd2b_22.pid
   ```

3. **TCP/IP Stacks**

   In this example started tasks SSHD2 and SSHD2B use TCP/IP stacks TCPIP and TCPIPB, respectively.

You can specify the TCP/IP stack using the environment variable _BPXK_SETIBMOPT_TRANSPORT. You can set it in the STDENV DD for BPXBATxx jobs, for example for SSHD2B:

```
//SSHD2B   PROC OPTS='-f /opt/tectia/etc/sshd2b_config' ❶
//TECTIA   EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//              PARM=('PGM /opt/tectia/sbin/sshd2 -F &OPTS')
//STDENV   DD  DSN=<HLQ>.V6611.PARMLIB(SSHENV),DISP=SHR
//         DD  DSN=<HLQ>.V6611.PARMLIB(TCPIPB),DISP=SHR ❷
//STDOUT   DD  SYSOUT=*
//*STDERR  DD  SYSOUT=*
//STDIN    DD  DUMMY
//         PEND
```

❶   The **sshd2** configuration file is specified with the -f option.

❷   The _BPXK_SETIBMOPT_TRANSPORT environment variable is set in <HLQ>.V6611.PARMLIB(TCPIPB):

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIPB
```

(Replace TCPIPB with the name of your TCP/IP stack.)

D.2 Running Two SOCKS Proxies on a Dual TCP/IP Stack

In this example we have two users, SSHSP and SSHSPB, and two started tasks, SSHSP and SSHSPB, for two Tectia SOCKS Proxies:

• **SSHSP:** this started task is assigned to user SSHSP and uses stack TCPIP:

```
//STDENV  DD  DSN=<HLQ>.V6611.PARMLIB(SSHENV),
//            DISP=SHR
//        DD  DSN=<HLQ>.V6611.PARMLIB(TCPIP),DISP=SHR ❶
```

❶   The TCP/IP stack name is specified using the _BPXK_SETIBMOPT_TRANSPORT environment variable that is set in <HLQ>.V6611.PARMLIB(TCPIP):

```
_BPXK_SETIBMOPT_TRANSPORT=TCPIP
```

• **SSHSPB:** this started task is assigned to user SSHSPB and uses stack TCPIPB:

```
//STDENV  DD  DSN=<HLQ>.V6611.PARMLIB(SSHENV),
//            DISP=SHR
//        DD  DSN=<HLQ>.V6611.PARMLIB(TCPIPB),DISP=SHR
```

The IP address of the SOCKS Proxy in stack TCPIP is 198.51.100.1, and the IP address of the SOCKS Proxy in stack TCPIPB is 198.51.100.2. The SOCKS Proxies are used to connect to remote servers at 203.0.113.1 and 203.0.113.2.
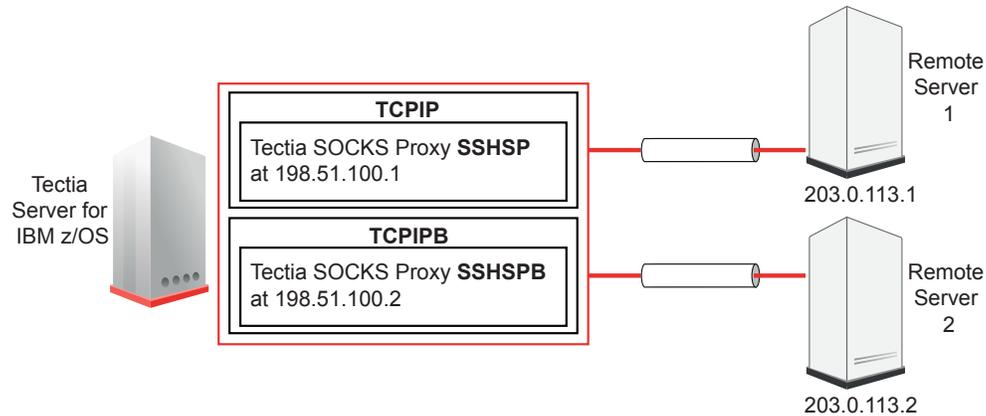
Figure D.1. Dual TCP/IP stack setup for Tectia SOCKS Proxy

You can run the two SOCKS proxies on a dual stack z/OS in two ways:

- Using two SOCKS Proxy configuration files with different network listeners (see Example D.1)

- Using one global SOCKS Proxy configuration file, and creating network listeners on both TCP/IP stacks' loopback address (see Example D.2). You can also use two separate SOCKS Proxy configuration files if you want to have different rules for the other stack connections.

Example D.1. Two configuration files with different network listeners

Add the following elements to the SSHSP configuration file (`/u/SSHSP/.ssh2/ssh-socks-proxy-config.xml`):

```
...
<profile name="dynamic-ftp"
         id="id1"
         host=""
         port="22"
         user="">
</profile>
...
<!-- SOCKS proxy needs its own listener for SOCKS. -->
<tunnel  type="socks-proxy"
         listen-address="198.51.100.1"
         listen-port="1080"
         dst-port="0"
         profile="" />
...
<rule    ip-address="203.0.113.1"
         ports="21"
         action="ftp-proxy"
         profile-id="id1"
         username-from-app="YES"
         hostname-from-app="YES"
         fallback-to-plain="NO" />
...
```

Add the following elements to the SSHSPB configuration file (`/u/SSHSPB/.ssh2/ssh-socks-proxy-config.xml`):

```
...
<profile name="dynamic-ftp"
         id="id1"
         host=""
         port="22"
         user="">
</profile>
...
<!-- SOCKS proxy needs its own listener for SOCKS. -->
<tunnel  type="socks-proxy"
         listen-address="198.51.100.2
         listen-port="1080"
         dst-port="0"
         profile="" />
...
<rule    ip-address="203.0.113.2"
         ports="21"
         action="ftp-proxy"
         profile-id="id1"
         username-from-app="YES"
         hostname-from-app="YES"
```

```
        fallback-to-plain="NO" />
...
```

Define the IP addresses of the SOCKS Proxies in the `socks.conf` file:

```
sockd @=198.51.100.1 203.0.113.1 255.255.255.0
sockd @=198.51.100.2 203.0.113.2 255.255.255.0
direct 0.0.0.0 0.0.0.0
```

Example D.2. Network listeners on TCP/IP stacks' loopback address

Add the following elements to the global SOCKS Proxy configuration file (`/opt/tectia/etc/ssh-socks-proxy-config.xml`):

```
...
<profile name="dynamic-ftp"
         id="id1"
         host=""
         port="22"
         user="">
</profile>
...
<!-- SOCKS proxy needs its own listener for SOCKS. -->
<tunnel  type="socks-proxy"
         listen-address="127.0.0.1"
         listen-port="1080"
         dst-port="0"
         profile="" />
...
<rule    ip-address="203.0.113.*"
         ports="21"
         action="ftp-proxy"
         profile-id="id1"
         username-from-app="YES"
         hostname-from-app="YES"
         fallback-to-plain="NO" />
...
```

Create listeners on both TCP/IP stacks' loopback address (`127.0.0.1`) in the `socks.conf` file:

```
sockd @=127.0.0.1 203.0.113.1 255.255.255.255
sockd @=127.0.0.1 203.0.113.2 255.255.255.0
direct 0.0.0.0 0.0.0.0
```

D.3 Connecting via Different TCP/IP Stacks with Tectia Clients

Note that running multiple Tectia clients simultaneously under the same user ID using different TCP/IP stacks will not work, because they use the same run-on-demand Connection Broker. The Connection Broker opens a socket on a particular TCP/IP stack; the clients (**sftpg3**, **scpg3**, **sshg3**) use local UNIX sockets to the Connection Broker.

If you want the clients to open connections to particular stacks, you have to run separate Connection Brokers on each stack, defined using the `_BPXK_SETIBMOPT_TRANSPORT` environment variable:

```
> env _BPXK_SETIBMOPT_TRANSPORT=TCPIPB ssh-broker-g3 -a /tmp/ssh-usernameB/ssh-broker
```

The **ssh-broker-g3** option `-a` starts the Connection Broker so that it listens to the Connection Broker connections on the local address `/tmp/ssh-usernameB/ssh-broker`. If the `/tmp/ssh-usernameB/` directory does not already exist, you have to first create it:

```
> mkdir /tmp/ssh-usernameB
```

> **ⓘ Note**
>
> If you want to set the TCP/IP stack permanently, you can do it by setting the `_BPXK_SETIBMOPT_TRANSPORT` environment variable to the stack name in the client's `.profile` file.

Use the `SSH_SECSH_BROKER` environment variable to point to the UNIX socket the Connection Broker is listening on:

```
> env SSH_SECSH_BROKER=/tmp/ssh-usernameB/ssh-broker sshg3 User@Server
```

You can use the **netstat** command to display all active connections for `TCPIPB`:

```
> netstat -a -p TCPIPB
```

Appendix E Console Messages

This appendix lists the console messages generated by Tectia Server for IBM z/OS.

The console messages generated by the SSH server (**sshd2**) and Certificate Validator (**ssh-certd**) are listed in Table E.1.

The console messages generated by the SOCKS Proxy (**ssh-socks-proxy**) are listed in Table E.2.

The messages are ordered by message code.

Table E.1. Console messages generated by the SSH server and Certificate Validator

Code	Message	Description
SSZ0001I	`<program_path_name>` `J=<job_name>,A=<ASID>,P=<PID>` ready	**Process started** Displayed when a new process is started.
SSZ0002W	`Command <command> unrecognised`	**Command unrecognised** Command `<command>` was not recognized.
SSZ0003I	`Command <command> accepted`	**Command accepted** Command `<command>` was accepted.
SSZ0004I	`Tectia Server for z/OS <version>`	**Version** Displayed as a response to the command VERSION.
SSZ0005I	`Dispatching signal to <pro-` `cess>=<PID> (<program_path_name>)`	**Stop / Restart / Reload** Displayed as a response to the commands STOP, RE-START and RELOAD. `<process>` is C or S, indicating a child or server process, respectively. `<PID>` is the identification number of the process the termination signal is dispatched to.
SSZ0006I	`Task <program_name> started`	**Task started** Displayed when the server is started using the START command.
SSZ0007W	`Command option <option> unrecog-` `nised`	**Command option unrecognised** Command option `<option>` was not recognized.
SSZ0008I	`VC branch: <branch>`	**Version control information: branch** Displayed as a response to the command VERSION.
SSZ0009I	`VC revision: <revision>`	**Version control information: revision** Displayed as a response to the command VERSION.

SSZ0010W	`Invalid access attempt. Dis-`
	`abled|Revoked|Undefined`
	`user=<user_ID> IP=<IP_address>`
		Invalid access attempt
		The SSH session will be disconnected during the authentication phase if the user ID used to log in is:
		
		• *Disabled* (user ID is configured with a UID value less than zero)
		
		• *Revoked* (user ID is marked as revoked in the RACF database)
		
		• *Undefined* (user ID is either not defined in the RACF database, or it does not have an OMVS segment)

Code	Message	Description
SSZ0011W	Maximum TcpListenRate reached	**Maximum TcpListenRate reached** The current TCP listen rate specified in sshd2_config TcpListenRate keyword exceeded.
SSZ0012W	CPU limit (*<current_usage>*, *<maximum_usage>*) exceeded (. getrlimit rc=*<return_value>*)	**CPU limit exceeded** The current CPU usage of sshd2 task exceeded the maximum usage. The task will be terminated. If system service getrlimit fails to the provide the current and maximum usage value, the return_value provided the possible failure causes.
SSZ0013W	ICSF Random number generator is not available	**ICSF Random number generator is not available** ICSF Random number generator service is not available. ICSF task has problem to provide random number service.
SSZ0014E	Error on listening tcp port: *<port_number>*	**Error on listening tcp port: port_number** There is error on listening tcp port number.
SSZ3050E	Error creating local stream directory	**Error creating local stream directory** The sft_server_g3 is not able to create a local stream file to communicate with sshd2 in /tmp/ssh-USERNAME

Table E.2. Console messages generated by the SOCKS Proxy

Code	Message	Description
SSZ3003I	`Command <command> accepted`	**Command accepted** Command *<command>* was accepted.
SSZ3004I	`Tectia Client for z/OS <version>`	**Version** Displayed as a response to the command VERSION.
SSZ3006I	`Task ssh-socks-proxy started`	**Task started** Displayed when the SOCKS Proxy is started.
SSZ3008I	`VC branch: <branch>`	**Version control information: branch** Displayed as a response to the command VERSION.
SSZ3009I	`VC revision: <revision>`	**Version control information: revision** Displayed as a response to the command VERSION.
SSZ3016I	`Broker running at '/tmp/ssh-SSH-SP/ssh-socks-proxy' (pid <pid>)`	**Address of the running SOCKS Proxy** Displayed as a response to the command STATUS.
SSZ3017I	`System configuration file: /opt/tectia/etc/ssh-socks-proxy-config.xml`	**Path to the SOCKS Proxy configuration file** Displayed as a response to the command STATUS.
SSZ3018I	`Agent socket path: /tmp/ssh-SSH-SP/ssh-socks-proxy-aa`	**Agent socket path** Displayed as a response to the command STATUS.
SSZ3019I	`Key upload log: /u/ssh-sp/.ssh2/keylog (not initialized)`	**Path to the key upload log** Displayed as a response to the command STATUS.
SSZ3023I	`<number> open connections, <number> clients`	**Number of open connections and clients** Displayed as a response to the command STATUS.
SSZ3025I	`Key exchange: <algorithm-list>`	**Supported key exchange algorithms** Displayed as a response to the command STATUS ALGORITHMS.
SSZ3026I	`Hostkey: <algorithm-list>`	**Supported host key algorithms** Displayed as a response to the command STATUS ALGORITHMS.
SSZ3027I	`Ciphers: <algorithm-list>`	**Supported ciphers** Displayed as a response to the command STATUS ALGORITHMS.
SSZ3028I	`MACs: <algorithm-list>`	**Supported MACs** Displayed as a response to the command STATUS ALGORITHMS.
SSZ3029I	`Compressions: <algorithm-list>`	**Supported compression algorithms** Displayed as a response to the command STATUS ALGORITHMS.

Code	Message	Description
SSZ3030I	User authentications: *<authentica-tion-methods>*	**Supported user authentication methods** Displayed as a response to the command STATUS ALGORITHMS.
SSZ3031I	Statistics: Total bytes sent: *<N>* Total bytes received: *<N>* Total connections opened: *<N>* Total clients opened: *<N>* Total client connections: *<N>*	**Statistics for the running SOCKS Proxy** Displayed as a response to the command STATUS.

Appendix F Log Messages

The log messages generated by Tectia Server for IBM z/OS are listed in the following sections based on their subject matter.

F.1 User Authentication - Common

Authentication failed

`<method-name>` `authentication failed. Login to account` `<login-name>` `not allowed or account non-existent.`

Level: warning

Facility: AUTH

The system administrator has denied logging in for the user, or the user is trying to log in using a non-existent account.

Authentication failed, user does not exist

`User` `<login-name>` `does not exist.`

Level: warning

Facility: AUTH

Authentication failed because the specified user account does not exist.

Root login denied for user

`root login denied for user '`<login-name>`'.`

Level: warning

Facility: AUTH

User tried to login as root (administrator), but system policy does not allow root (administrator) logins at all or with password authentication.

Authentication failed, system policy does not allow user's login

`User <login-name>'s login is not allowed due to system policy`
Level: warning

Facility: AUTH

Authentication failed because the system policy does not allow the user to log in. The account may be locked or expired, for example.

Authentication failed, user name is on `DenyUsers` list

`User <login-name> is denied login because username matched with the deny list.`
Level: warning

Facility: AUTH

Authentication failed because the specified user name is included in the server's denied users list.

Authentication failed, user name not on `AllowUsers` list

`User <login-name> is denied login because allow list exists, and user name was not matched.`
Level: warning

Facility: AUTH

Authentication failed because the specified user name is not included in the server's allowed users list.

Authentication not allowed from a host

`<method-name> authentication failed. Connection from <hostname> denied. Authentication as user <login-name> was attempted.`
Level: warning

Facility: AUTH

The server has determined that the user is not allowed to login from the current host.

Authentication not allowed, unable to reverse map host name

`Login not allowed because host's IP address (<ip-address>) could not be mapped to a hostname and reverse mapping is required.`
Level: warning

Facility: AUTH

The server could not map the connecting host's IP address to a host name and the configuration option `RequireReverseMapping` is on.

Authentication not allowed, host is on the `DenyHosts` list

```
Login not allowed because the connecting host (<hostname>) is on the server's deny list.
```
Level: warning

Facility: AUTH

The connecting host is on the server's denied hosts list, so the authentication is not allowed.

Authentication not allowed, host is not on the `AllowHosts` list

```
Login not allowed because the connecting host (<hostname>) is not on the server's allow
list.
```
Level: warning

Facility: AUTH

The connecting host is not on the server's allowed hosts list, so the authentication is not allowed.

F.2 User Authentication - Host-Based

Using host-based authentication denied for a host

```
Use of <filename> denied for <hostname>
```
Level: warning

Facility: AUTH

Use of host-based authentication is denied for the host in server configuration.

Configuration files missing for host-based authentication

```
hostbased-authentication (rhosts) refused for <login-name> no .shosts or .rhosts files
and no system-wide files (ie: /etc/shosts.equiv)
```
Level: warning

Facility: AUTH

The required files for host-based authentication are missing.

Home directory missing in host-based authentication

```
hostbased-authentication (rhosts) refused for <login-name>: no home directory <directory>
```
Level: warning

Facility: AUTH

The user's home directory does not exist.

Home directory ownership or permissions invalid in host-based authentication

```
hostbased-authentication (rhosts) refused for <login-name>: bad ownership or modes for
home directory.
```
Level: warning

Facility: AUTH

If the `StrictModes` option for the server is switched on, then this message means that the user's home directory
is not owned by either the user or the root, or that its permission bits are not correct.

`.rhosts` file ownership or permissions invalid in host-based authentication

```
hostbased-authentication (rhosts) refused for <login-name>: bad modes for <file>
```
Level: warning

Facility: AUTH

If the `StrictModes` option for the server is switched on, then this message means that the user's `.rhosts` file
is not owned either by the user or by the root, or that its permission bits are not correct.

Host-based authentication refused for user

```
hostbased-authentication (rhosts) refused: client user <login-name>, server user <login-
name>, client host <hostname>.
```
Level: warning

Facility: AUTH

The host-based authentication attempt has failed.

Client's host certificate is not valid in host-based authentication

```
Hostbased: User <login-name>'s client host certificate not valid, error: <error>.
```
Level: warning

Facility: AUTH

The host certificate sent by the client during host-based authentication could not be successfully validated.

Certificate does not contain the client host name in host-based authentication

```
Certificate does not contain client hostname <hostname>.
```
Level: warning

Facility: AUTH

The host certificate sent by the client during host-based authentication does not contain the client's host name.

Certificate could not be decoded in host-based authentication

```
Received certificate could not be decoded
```
Level: notice

Facility: AUTH

The host certificate sent by the client during host-based authentication could not be decoded.

Public key could not be extracted from the received certificate in host-based authentication

```
Could not extract public key from received certificate
```
Level: notice

Facility: AUTH

Extracting the public key from the received certificate failed during host-based authentication.

Cryptographic operation failed in host-based authentication

```
Hostbased: Crypto operation failed for public key (for user <login-name>).
```
Level: warning

Facility: AUTH

Changing the hashing scheme in the public key failed in host-based authentication.

Verifying the signature failed in host-based authentication

```
Hostbased authentication failed for <login-name>.
```
Level: warning

Facility: AUTH

Verifying the signature failed in host-based authentication.

Host-based authentication successful

```
Hostbased authentication for user <login-name> accepted.
```
Level: notice

Facility: AUTH

Authentication using the host-based authentication method succeeded for the user.

Could not decode packet in host-based authentication

```
Error decoding "hostbased" packet.
```
Level: warning

Facility: AUTH

Decoding the host-based authentication protocol packet failed.

The host name given by the client does not match the one given by the client in host-based authentication

```
Client gave us a hostname (<hostname>) which doesn't match the one we got from DNS
(<hostname>) (sending failure)
```
Level: warning

Facility: AUTH

The server has been configured to use strict host name checking and the host name obtained using reverse DNS lookup does not match the one in the host-based authentication packet.

The host name given by client does not match the one given by the client in host-based authentication, ignored

```
Client gave us a hostname (<hostname>) which doesn't match the one we got from DNS
(<hostname>) (trusting that client is valid, if signature verification succeeds)
```
Level: warning

Facility: AUTH

The host name obtained using reverse DNS lookup does not match the one in the host-based authentication packet, but the server does not require a match.

Invalid packet received in host-based authentication

```
Invalid packet. (extra data at end)
```
Level: warning

Facility: AUTH

The received packet in host-based authentication contains extra data and will be discarded as invalid.

Client is using a public-key algorithm not supported by server in host-based authentication

```
Client's public key algorithms are not supported by us. (client sent <algorithm-name>)
```
Level: warning

Facility: AUTH

The public key sent by the client in host-based authentication uses a public key algorithm not supported by the server.

User's `.ssh2` directory is missing in host-based authentication

```
Couldn't find or create user <login-name>'s .ssh2 directory.
```

Level: warning

Facility: AUTH

The user's `.ssh2` directory could not be found or created in host-based authentication.

Received host key and stored host key differ in host-based authentication

`The public key stored in <filename> and the one given by the client were different.`
Level: warning

Facility: AUTH

The host key sent by the client and the one stored in the known host keys directory differ in host-based authentication.

Invalid client host key in host-based authentication

`Importing client pubkey failed.`
Level: warning

Facility: AUTH

There was an error when decoding the host key sent by the client during host-based authentication.

F.3 User Authentication - Keyboard-Interactive Password

Login with empty password denied for user

`kbd-int: passwd: login with empty password denied for user '<login-name>'.`
Level: warning

Facility: AUTH

The server policy forbids login with empty password.

Denied login with too long a password

`kbd-int: passwd: user '<login-name>' tried to login with too long password (256)`
Level: warning

Facility: AUTH

The server does not allow logins with passwords longer than 256 characters.

Kerberos password accepted

`kbd-int: passwd: kerberos password accepted for user <login-name> (<kerberos-name>).`

Level: notice

Facility: AUTH

The server accepted the user's Kerberos password in Keyboard- Interactive password authentication.

Password not accepted

```
kbd-int: passwd: wrong password given for user '<login-name>'.
```
Level: warning

Facility: AUTH

The password supplied by the client was invalid.

Password must be changed

```
kbd-int: passwd: user '<login-name>' forced to change password.
```
Level: informational

Facility: AUTH

The server has determined that the user's password must be changed.

Error during password changing

```
kbd-int: passwd: user '<login-name>': <error-message>
```
Level: warning

Facility: AUTH

There was an error during password changing.

Password change done

```
kbd-int: passwd: user '<login-name>' passwd change [successful|failed]
```
Level: notice

Facility: AUTH

Password change resulted in success or failure.

F.4 User Authentication - Keyboard-Interactive

Client sent fewer responses than server sent requests

```
auth-kbd-int: client sent us fewer responses than we sent requests (sent: <sent-requests>,
recv: <received-responses>).
```

Level: error

Facility: AUTH

Protocol error: the client sent fewer responses than the server sent requests.

Login denied, faking transaction

```
auth-kbd-int: User '<login-name>' login denied, faking real transaction.
```
Level: warning

Facility: AUTH

The user is not allowed to log in, but the rest of the transaction proceeds normally, except that all results are marked as failures.

F.5 User Authentication - Password

User not allowed to log in after password change

```
User '<login-name>' not allowed to log in after password change (strange).
```
Level: warning

Facility: AUTH

For some reason, the system denies login for the user after a mandatory password change.

Password change required after password change

```
User '<login-name>' password must be changed after password change (strange).
```
Level: warning

Facility: AUTH

For some reason, the system requires the user to change their password even after a mandatory password change.

Password change failed

```
Password change for user '<login-name>' failed (reason: <error-message>).
```
Level: warning

Facility: AUTH

Password change failed.

Password change succeeded

Password change for user '*<login-name>*' succeeded.

Level: notice

Facility: AUTH

Password change succeeded.

Error during password change

passwd_change: error_msg: '*<error-message>*'

Level: warning

Facility: AUTH

An error happened during password changing.

The server does not allow root logins

root login denied for user '*<login-name>*'.

Level: warning

Facility: AUTH

The server's configuration does not allow root (administrator) logins or does not allow root logins with a password.

The server does not allow logins with empty passwords

attempt to login as *<login-name>* with empty password denied.

Level: warning

Facility: AUTH

The server's configuration does not allow logins with empty passwords.

The server does not allow logins with passwords longer than 256 characters

attempt to login as *<login-name>* with password longer than 256 characters denied.

Level: warning

Facility: AUTH

The server does not allow logins with passwords longer than 256 characters.

Kerberos password accepted

Kerberos password accepted for user *<login-name>* (*<kerberos-name>*).

Level: notice

Facility: AUTH

The server accepted the Kerberos password for the user.

Password accepted

`User <login-name>'s local password accepted.`
Level: notice

Facility: AUTH

The server accepted the user's local password.

Password not accepted

`Wrong password given for user '<login-name>'.`
Level: warning

Facility: AUTH

The password supplied by the client was invalid.

Password authentication accepted

`Password authentication for user <login-name> accepted.`
Level: notice

Facility: AUTH

The client passed the password authentication, but the server still needs to check whether the password needs to be changed.

Badly formatted password change request

`Badly formatted password change request, denying already successful authentication.`
Level: notice

Facility: AUTH

The server received a badly formatted password change request, and the authentication will be noted as failed.

Changing user's password

`Changing <login-name>'s password.`
Level: notice

Facility: AUTH

The server is now trying to change the user's password.

Password change needed for the user

```
User <login-name> forced to change password.
```
Level: informational

Facility: AUTH

The server has determined that the user's password must be changed.

User exceeds maximum number of password guesses

```
PasswordGuesses exceeded.
```
Level: warning

Facility: AUTH

User exceeded maximum number of password guesses.

F.6 User Authentication - Public Key

Certificate not authorized for login

```
Authorization check for user <login-name>'s certificate rejected[, reason: <error-message>].
```
Level: warning

Facility: AUTH

The server's authorization check for the certificate produced a negative result, meaning that public-key authentication with this certificate is denied.

Error in public key decoded from certificate

```
Couldn't get info from certificate public key used by user <login-name>.
```
Level: warning

Facility: AUTH

After extracting the public key from the received certificate, the server could not get the public key information from the public key.

Key length too small in certificate

```
Use of certificate for user <login-name> is not allowed, because public key is too small
(pk size <key-size>, min size <minimum-size>).
```
Level: warning

Facility: AUTH

The server has rejected the authentication using the certificate sent by the client, because the public key in the certificate is too short.

Key length too large in certificate

```
Use of certificate for user <login-name> is not allowed, because public key is too large
(pk size <key-size>, max size <maximum-size>).
```
Level: warning

Facility: AUTH

The server has rejected the authentication using the certificate sent by the client because the public key in the certificate is too long.

Signature check failed

```
Signature verify failed for certificate from user <login-name>.
```
Level: warning

Facility: AUTH

The signature verification failed for the user in certificate authentication.

Certificate authentication accepted

```
Certificate authentication for user <login-name> accepted.
```
Level: notice

Facility: AUTH

The server has accepted the certificate authentication for the user.

Certificate authentication rejected

```
Certificate authentication for user <login-name> rejected.
```
Level: warning

Facility: AUTH

The server has rejected the certificate authentication for the user.

Public key options prohibit using the key

```
Use of public key <key-name> is not allowed in public key options for the host the user
<login-name> is logging in from.
```
Level: notice

Facility: AUTH

Using the public key for the user and host has been denied based on the public key options set on the server for the key.

Error in public key

```
Couldn't get info from public key <key-name> used by user <login-name>.
```
Level: warning

Facility: AUTH

The server could not get the public key information from the public key.

Key length too small

```
Use of public key <key-name> by user <login-name> is not allowed, because public key is too
small (pk size <key-size>, min size <minimum-size>).
```
Level: warning

Facility: AUTH

The server has rejected the authentication using the public key sent by the client because the key is too short.

Key length too large

```
Use of public key <key-name> by user <login-name> is not allowed, because public key is too
large (pk size <key-size>, max size <maximum-size>).
```
Level: warning

Facility: AUTH

The server has rejected the authentication using the public key sent by the client because the key is too long.

Cryptographic operation failed for RSA key

```
Crypto operation failed for RSA key of user <login-name>.
```
Level: warning

Facility: AUTH

A cryptographic operation failed for an RSA key.

Root login permitted for forced command

```
root login permitted for forced command.
```
Level: notice

Facility: AUTH

The server has been configured to not allow root (administrator) logins, but will allow forced commands as root.

Public-key authentication accepted

```
Public key authentication for user <login-name> accepted.
```
Level: notice

Facility: AUTH

The server has accepted the public-key authentication for the user.

Bad packet when verifying key

```
got bad packet when verifying user <login-name>'s publickey.
```
Level: warning

Facility: AUTH

The server received an invalid packet when trying to verify the user's public key.

Public key is authorized

```
Public key <filename> authorized for user <login-name>, verifying signature.
```
Level: notice

Facility: AUTH

The server has determined that the public key supplied by the client is authorized for logging in, and will now verify the signature supplied by the client.

User's public key logged with fingerprint

```
User authorized by public key: <key-comment>, fingerprint: <key-fingerprint>
```
Level: informational

Facility: AUTH

The server logs the user's public key fingerprint.

User's public key logged, fingerprint not created

```
User authorized by public key (could not create a fingerprint): <key-comment>: <key-data>
```
Level: informational

Facility: AUTH

The server was unable to create the user's public key fingerprint when logging the user's public key.

F.7 Certificate-Specific Code

Certificate verification failed

```
Certificate verification failed: search-state = {<search-state>}
```
Level: informational

Facility: AUTH

Verifying a certificate failed in user or host authentication. `<search-state>` contains the search status bits that give information about what went wrong.

Opening listener succeeds

```
Opening listener at <path> succeeded.
```
Level: informational

Facility: AUTH

Listener was successfully opened.

Opening listener fails

```
Opening listener at <path> failed.
```
Level: informational

Facility: AUTH

Opening listener failed.

F.8 Agent Forwarding

Agent forwarding requested

```
Agent forwarding requested, creating a listener.
```
Level: informational

Facility: DAEMON

The client has requested agent forwarding, and the server is creating a listener for it.

Agent forwarding request has been denied

```
Agent forwarding request denied in configuration.
```
Level: informational

Facility: DAEMON

The client has requested agent forwarding, but the server configuration forbids it.

Forwarding an agent connection

```
Opening an agent forwarding channel
```
Level: informational

Facility: DAEMON

The server has received an agent connection in the agent forwarding listener, and is opening an agent forwarding channel to the client.

Closing an agent connection

```
Closing an agent forwarding channel
```
Level: informational

Facility: DAEMON

The agent forwarding channel from the server to the client has been closed.

Agent forwarding request fails

```
Agent forwarding request failed as user <username> is denied terminal access in the server
configuration
```
Level: informational

Facility: DAEMON

The user is denied terminal access in the server configuration, resulting in agent forwarding request failing.

F.9 Session Channels

Session channel open request received

```
Received a channel open request, type <channel-type>, channel id <channel-number>
```
Level: informational

Facility: DAEMON

Received a request to open a session channel of the specified type with specified channel ID.

Closing session channel

```
Destroying session channel <channel-number>
```
Level: informational

Facility: DAEMON

The server is destroying the session channel with the specified ID number.

Session channel extension request received

```
Received a session channel extension request of type <extension-type> for channel number
<channel-number>
```
Level: informational

Facility: DAEMON

Received a session channel extension request of the specified type for the channel with specified ID number.

Setting environment variable denied

```
User tried to set environment variable '<environment-variable>' to '<variable-value>' in "<fi-
lename>", but was denied by policy.
```
Level: notice

Facility: DAEMON

Setting the environment variable denied by server policy.

Opening log for internal SFTP server

```
Opening log for internal sftp-server.
```
Level: informational

Facility: DAEMON

The server opened the log for the internal SFTP server.

Chrooting user

```
User '<login-name>' will be chrooted to directory '<directory>'.
```
Level: informational

Facility: DAEMON

The server is chrooting the user into a directory.

Changed to run on user privileges

```
Now running on <login-name>'s privileges.
```
Level: informational

Facility: DAEMON

The server has dropped root (administrator) privileges and is now running on user's privileges.

Setting environment variable denied by public key options

User tried to set environment variable '`<environment-variable>`' to '`<variable-value>`' in public key options, but was denied by policy.

Level: notice

Facility: DAEMON

Setting the environment variable denied by public key options.

Starting internal SFTP server

Starting internal sftp-server for user '`<login-name>`', sshd2 PID [`<pid-number>`].

Level: notice

Facility: LOCAL7

The internal SFTP server is starting.

Stopping internal SFTP server

Stopping internal sftp-server for user '`<login-name>`', sshd2 PID [`<pid-number>`].

Level: notice

Facility: LOCAL7

The internal SFTP server is stopping.

Bad data received in environment variable setting

Bad data on ssh_channel_request_env().

Level: warning

Facility: DAEMON

Decoding an environment variable setting request failed in the ssh_channel_request_env function.

Invalid values received in environment variable setting

Invalid values on request to set environment variable: name='`<variable-name>`',value='`<variable-value>`'.

Level: warning

Facility: DAEMON

The environment variable setting request contained invalid values.

Environment variable setting forbidden by policy

Client tried to set environment variable '`<variable-name>`' to '`<variable-value>`', but it is forbidden by policy.

Level: notice

Facility: DAEMON

The environment variable setting failed because of server policy.

Executing session for user fails

```
Executing session for user '<username>' failed. Freeing confidential data failed, exiting.
```
Level: error

Facility: DAEMON

Executing session for user, and freeing confidential data both fail, resulting in exiting.

Forced command fails

```
Forced command failed as user <username> is denied terminal access in the server configur-
ation
```
Level: informational

Facility: DAEMON

User is denied terminal access in the server configuration, resulting in forced command failing.

Pty request fails

```
Pty request failed as user <username> is denied terminal access in the server configuration
```
Level: informational

Facility: DAEMON

User is denied terminal.

Shell request fails

```
Shell request failed as user <username> is denied terminal access in the server configur-
ation
```
Level: informational

Facility: DAEMON

User is denied terminal access in the server configuration, resulting in shell request failing.

Exec request fails

```
Exec request failed as user <username> is denied terminal access in the server configuration
```
Level: informational

Facility: DAEMON

User is denied terminal access in the server configuration, resulting in exec request failing.

F.10 SSH1 Agent Forwarding

SSH1 agent forwarding denied

```
SSH1 agent forwarding request denied in configuration.
```
Level: informational

Facility: DAEMON

SSH1 agent forwarding is denied by the server configuration.

Creating SSH1 agent listener

```
SSH1 agent forwarding requested, creating a listener.
```
Level: informational

Facility: DAEMON

SSH1 agent forwarding allowed, creating an agent listener.

Opening an SSH1 agent forwarding channel

```
Opening an SSH1 agent forwarding channel
```
Level: informational

Facility: DAEMON

SSH1 agent connection received, opening a forwarding channel.

Closing an SSH1 agent forwarding channel

```
Closing an ssh1 agent forwarding channel
```
Level: informational

Facility: DAEMON

Closing a SSH1 agent forwarding channel.

SSH1 agent forwarding fails

```
SSH1 agent forwarding request failed as user <username> is denied terminal access in the
server configuration
```
Level: informational

Facility: DAEMON

The user is denied terminal access in the server configuration, resulting in SSH1 agent forwarding request failing.

F.11 Port Forwarding

Direct TCP port forwarding request denied

`Direct TCP/IP forwarding request (<host>:<port>) denied for user in configuration.`
Level: informational

Facility: DAEMON

A direct TCP port forwarding request has been denied because of system policy.

Direct TCP port forwarding forbidden

`Direct TCP/IP forwarding request denied in configuration.`
Level: informational

Facility: DAEMON

A direct TCP port forwarding request was denied because the system policy forbids TCP port forwarding.

Connection received to forwarded port

`Connection to forwarded port <forwarded-port> from <host>:<port>`
Level: informational

Facility: DAEMON

The forwarded port received a connection.

Remote port forwarding connection received

`Remote fwd connect from <connection-originator> to local port <port>`
Level: informational

Facility: DAEMON

A remote port forwarding request was received.

Remote listener creation failed

`Creating remote listener for <address>:<port> failed.`
Level: notice

Facility: DAEMON

A remote listener creation failed.

Port set up for remote forwarding

```
Port <port> set up for remote forwarding.
```
Level: informational

Facility: DAEMON

A port was successfully set up for remote forwarding.

Received a remote forwarding request

```
Remote TCP/IP forwarding request received from host "<host>", by authenticated user
"<login-name>".
```
Level: informational

Facility: DAEMON

Received a remote forwarding request.

Forwarding a privileged port

```
Privileged user "<login-name>" forwarding a privileged port.
```
Level: notice

Facility: DAEMON

Forwarding a privileged port (under 1024) for a privileged user.

Remote TCP port forwarding forbidden

```
Remote TCP/IP forwarding request denied in configuration.
```
Level: informational

Facility: DAEMON

A remote TCP port forwarding request was denied because the system policy forbids TCP port forwarding.

Remote TCP port forwarding request denied

```
Remote TCP/IP forwarding (<host>:<port>) request denied for user in configuration.
```
Level: informational

Facility: DAEMON

A remote TCP port forwarding request has been denied because of system policy.

Connection to direct forwarding received

```
direct fwd connect from <connection-originator> to local port <port>
```
Level: informational

Facility: DAEMON

Received a connection to forwarded port.

F.12 Common Code

Disconnected

```
[Local|Remote host] disconnected: <reason-message>
```
Level: informational

Facility: DAEMON

The connection was disconnected by the local or the remote end for the reason given.

F.13 Host Key I/O

PIN request received

```
PIN requested, PIN label <pin-label> at keypath <keypath>, ignoring.
```
Level: warning

Facility: DAEMON

A PIN request was received when reading host keys, and since the user is not available at this point, the request
is ignored.

F.14 Cryptography Support

Random Number Generation

```
Support for random number generation: [/dev/random|software]
```
Level: informational

Facility: DAEMON

Cryptography support for random number generation: '/dev/random' for hardware support, or 'software' for
software support.

Cipher

```
Support for cipher <cipher-name>: [ICSF|software]
```
Level: informational

Facility: DAEMON

Cryptography support for a given cipher: 'ICSF' for hardware support, or 'software' for software support.

MAC

```
Support for MAC <MAC-name>: [ICSF|software]
```
Level: informational

Facility: DAEMON

Cryptography support for a given MAC algorithm: 'ICSF' for hardware support, or 'software' for software support.

F.15 General Server Log Messages

Disconnected: Host not allowed to connect

```
Disallowed connect from denied host. '<message>'
```
Level: warning

Facility: DAEMON

The server has disconnected the client because the client is not allowed to connect.

Disconnected: Protocol error

```
Protocol error in [local|remote]: '<message>'
```
Level: warning

Facility: DAEMON

Disconnecting because of a protocol error in local or remote end.

Disconnected: Key exchange failed

```
Key exchange failed in [local|remote]: '<message>'
```
Level: warning

Facility: DAEMON

Disconnecting because key exchange failed in local or remote end.

Disconnected: Illegal code

```
RESERVED (this is an illegal code) in [local|remote]: '<message>'
```
Level: warning

Facility: DAEMON

Disconnecting because of illegal code.

Disconnected: MAC failed

```
MAC failed in [local|remote], disconnecting: '<message>'
```
Level: warning

Facility: DAEMON

Disconnecting because MAC failed in local or remote end.

Disconnected: Compression error

```
compression error in [local|remote], disconnecting: '<message>'
```
Level: warning

Facility: DAEMON

Disconnecting because of a compression error.

Disconnected: Service not available

```
service not available: '<message>'
```
Level: warning

Facility: DAEMON

Disconnecting because a requested service was not available.

Disconnected: Protocol version not supported

```
protocol version not supported in [local|remote]: '<message>'
```
Level: informational

Facility: DAEMON

Disconnecting because the protocol version was not supported.

Disconnected: Host key not verified

```
host key not verifiable in [local|remote]: '<message>'
```
Level: warning

Facility: DAEMON

Disconnecting because the host key could not be verified.

Disconnected: Connection was lost

```
connection lost: '<message>'
```
Level: informational

Facility: DAEMON

Disconnecting because the connection was lost.

Disconnected: Disconnected by application

```
disconnected by application in [local|remote]: '<message>'
```
Level: informational

Facility: DAEMON

Disconnected by application.

Disconnected: Too many connections

```
too many connections : '<message>'
```
Level: informational

Facility: DAEMON

Disconnected because the internal connection limit was exceeded.

Disconnected: User authentication failed

```
User authentication failed: '<message>'
```
Level: warning

Facility: DAEMON

Disconnected because the user authentication failed. This message should only occur when the other end is of version 2.0.13 or earlier.

Disconnected: Authentication canceled by user

```
authentication cancelled by user: '<message>'
```
Level: informational

Facility: DAEMON

Disconnected because the authentication was canceled by the user.

Disconnected: No more authentication methods

```
no more authentication methods on [local|remote]: '<message>'
```
Level: informational

Facility: DAEMON

Disconnected because of running out of authentication methods. This generally means that the client failed in all the authentication methods available on the server.

Disconnected: Illegal user name

```
illegal user name : '<message>'
```
Level: informational

Facility: DAEMON

Disconnected because the user name was illegal.

Disconnected: Unknown reason

```
Unknown reason code '<reason-code>' for disconnect. msg: '<message>'
```
Level: error

Facility: DAEMON

Disconnected for an unknown reason.

Idle timeout exceeded

```
Idle timeout exceeded.
```
Level: warning

Facility: DAEMON

The idle timeout limit was exceeded.

Remote protocol version string received

```
Remote protocol version: <version-string>
```
Level: informational

Facility: DAEMON

The server has received the remote protocol version string from the client.

User authenticated

```
User <login-name> (uid <uid-number>), coming from <host>, authenticated.
```
Level: notice

Facility: DAEMON

A general message stating that a user has been successfully authenticated.

User authenticated, root login

`ROOT LOGIN: User <login-name> (uid <uid-number>), coming from <host>, authenticated.`
Level: notice

Facility: DAEMON

General message stating that a superuser has been successfully authenticated.

User authentication failed

`Authentication failed for user <login-name>, coming from <host>.`
Level: warning

Facility: DAEMON

A user tried to log in, but user authentication failed.

User logged out

`Logout for user <login-name>.`
Level: informational

Facility: DAEMON

The specified user has now logged out or has been logged out.

Login grace time exceeded

`LoginGraceTime exceeded.`
Level: warning

Facility: DAEMON

The `LoginGraceTime` value was exceeded. This means that the client did not succeed in logging in during the time period defined by the configuration option `LoginGraceTime`.

Connection received

`connection from "<host>" (listen port: <listen-port>)`
Level: informational

Facility: DAEMON

The server has received a connection from a host.

Connection received in interface

```
connection from "<host>" (listen iface: <address>:<port>)
```
Level: informational

Facility: DAEMON

The server has received a connection from a host to a port on specific interface.

Refused connection: too many open connections

```
Refusing connection from "<host>". Too many open connections (max <maximum-connections>, now
open <open-connections>).
```
Level: warning

Facility: DAEMON

The server has refused a connection because it would have otherwise exceeded the limit of simultaneously open connections.

Denied connection because of `tcp_wrappers`

```
Denied connection from <connection-originator> by tcp wrappers.
```
Level: warning

Facility: DAEMON

The server has refused a connection because it was denied by `tcp_wrappers`.

Forking for new connection failed

```
Forking a server for a new connection failed.
```
Level: warning

Facility: DAEMON

The server tried to fork to handle a new connection, but the forking failed.

Server warning

```
WARNING: <message>
```
Level: warning

Facility: DAEMON

A warning was received from the server.

Fatal server error

```
FATAL ERROR: <message>
```
Level: error

Facility: DAEMON

The server has encountered a fatal error condition.

`SIGHUP` handler received an invalid signal

```
Server's SIGHUP handler received an illegal signal.
```
Level: warning

Facility: DAEMON

Invalid signal received by SIGHUP-handler.

License file error

```
License file error: <message>
```
Level: error

Facility: DAEMON

The server encountered an error when checking the license file.

Server restarting

```
restarting...
```
Level: warning

Facility: DAEMON

The server is restarting.

Server starting in `inetd` mode

```
Starting daemon in inetd mode.
```
Level: warning

Facility: DAEMON

The server is starting in `inetd` mode.

Server created a listener

```
Listener created on <address>:<port>.
```
Level: warning

Facility: DAEMON

The server has successfully created a listener.

Server created an UDP listener

```
UDP Listener created on <address>:<port>.
```
Level: warning

Facility: DAEMON

The server has successfully created an UDP listener.

`setsid` **failed**

```
setsid: <error-message>
```
Level: notice

Facility: DAEMON

A call to setsid failed in the server.

Server running

```
Daemon is running.
```
Level: warning

Facility: DAEMON

The server process has successfully performed all initialization operations and is now listening for connections.

SIG(INT|TERM) handler received an illegal signal

```
Invalid signal received by SIG(INT|TERM)-handler.
```
Level: warning

Facility: DAEMON

Server's SIG(INT|TERM) handler received an illegal signal.

SIGUSR1 handler received an illegal signal

```
Invalid signal received by SIGUSR1-handler.
```
Level: warning

Facility: DAEMON

Server's SIGUSR1 handler received an illegal signal.

F.16 SFTP

File closed

```
File '<filename>' closed (written: <bytes-written>, read: <bytes-read>).
```

Level: notice

Facility: LOCAL7

A file has been closed. This is the only log message for a completed file transfer. Typically the whole file is read or written, but this is not the only possibility. This log message will report the sum of all reads and the sum of all writes since opening the file. (Note: Even the same client can open a file multiple times, which will result in multiple closes, each with a possibly different number of read and written bytes.)

Bad message received

```
Received bad message <message-number>.
```
Level: error

Facility: LOCAL7

The protocol code received a bad message from the client.

Got wrong message before receiving `INIT`

```
Got message type <message-type> before receiving INIT. This is a protocol error, destroying
the file-server.
```
Level: error

Facility: LOCAL7

A protocol error occurred, and the file server is being terminated.

Received `SSH_FXP_INIT`

```
Received SSH_FXP_INIT
```
Level: notice

Facility: LOCAL7

The file server received the `SSH_FXP_INIT` message from the client.

Could not decode request ID

```
Couldn't decode request id. Destroying file-server.
```
Level: error

Facility: LOCAL7

A protocol error took place, and the file server is being terminated.

Received `SSH_FXP_OPEN`

```
Received SSH_FXP_OPEN
```

Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_OPEN message from the client.

Received **SSH_FXP_CLOSE**

Received SSH_FXP_CLOSE
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_CLOSE message from the client.

Received **SSH_FXP_STAT**

Received SSH_FXP_STAT
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_STAT message from the client.

Received **SSH_FXP_LSTAT**

Received SSH_FXP_LSTAT
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_LSTAT message from the client.

Received **SSH_FXP_FSTAT**

Received SSH_FXP_FSTAT
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_FSTAT message from the client.

Received **SSH_FXP_SETSTAT**

Received SSH_FXP_SETSTAT
Level: notice

Facility: LOCAL7

The file server received an SSH_FXP_SETSTAT message from the client.

Received `SSH_FXP_FSETSTAT`

Received SSH_FXP_FSETSTAT
Level: notice

Facility: LOCAL7

The file server received an SSH_FXP_FSETSTAT message from the client.

Received `SSH_FXP_OPENDIR`

Received SSH_FXP_OPENDIR
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_OPENDIR message from the client.

Received `SSH_FXP_READDIR`

Received SSH_FXP_READDIR
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_READDIR message from the client.

Received `SSH_FXP_REMOVE`

Received SSH_FXP_REMOVE
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_REMOVE message from the client.

Received `SSH_FXP_MKDIR`

Received SSH_FXP_MKDIR
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_MKDIR message from the client.

Received `SSH_FXP_RMDIR`

Received SSH_FXP_RMDIR
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_RMDIR message from the client.

Received **SSH_FXP_REALPATH**

Received SSH_FXP_REALPATH
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_REALPATH message from the client.

Received **SSH_FXP_RENAME**

Received SSH_FXP_RENAME
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_RENAME message from the client.

Received **SSH_FXP_READLINK**

Received SSH_FXP_READLINK
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_READLINK message from the client.

Received **SSH_FXP_SYMLINK**

Received SSH_FXP_SYMLINK
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_SYMLINK message from the client.

Received **SSH_FXP_EXTENDED**

Received SSH_FXP_EXTENDED
Level: informational

Facility: LOCAL7

The file server received an SSH_FXP_EXTENDED message from the client.

Received bad **SSH_FXP_INIT**

`Received bad SSH_FXP_INIT`

Level: error

Facility: LOCAL7

The server received a bad `SSH_FXP_INIT` message.

Negotiated protocol version

`Negotiated version <version-number>.`

Level: notice

Facility: LOCAL7

The server has negotiated to use the specified protocol version with the client.

Uploading file

`Uploading '<filename>' (flags [append] [trunc] [creat] [excl])`

Level: notice

Facility: LOCAL7

The server has received a write request for the specified file with the specified flags.

Downloading file

`Downloading '<filename>'`

Level: notice

Facility: LOCAL7

The server has received a read request for the specified file.

Failed to open file

`Failed to open file '<filename>'. Error <error-number>.`

Level: warning

Facility: LOCAL7

The server failed to open the specified file.

Closed file

`[ Closed | Received error when closing ] file '<filename>' (handle=<handle>)`

Level: notice

Facility: LOCAL7

The server closed the file, possibly with an error during the close.

Statting file

```
Statting file '<filename>'
```
Level: informational

Facility: LOCAL7

The server is statting the specified file.

Lstatting file

```
Lstatting file '<filename>'
```
Level: informational

Facility: LOCAL7

The server is lstatting the specified file.

Fstatting file

```
Fstatting file '<filename>'
```
Level: informational

Facility: LOCAL7

The server is fstatting the specified file.

Setstatting file

```
Setstat file '<filename>' ([size] [uidgid] [perms] [acmodtime])
```
Level: notice

Facility: LOCAL7

The server is setstatting the attributes of the specified file.

Fsetstatting file

```
Fsetstat file '<filename>' (handle=<handle>) ([size] [uidgid] [perms] [acmodtime])
```
Level: notice

Facility: LOCAL7

The server is fsetstatting the attributes of the specified file.

Opening directory

```
Opening directory '<filename>'
```

Level: notice

Facility: LOCAL7

The server is opening the specified directory.

Failed to open directory

```
Failed to open dir '<filename>'. Error <error-number>.
```
Level: warning

Facility: LOCAL7

The server failed to open the specified directory.

Readdir on directory

```
Readdir on directory '<filename>' (handle=<handle>)
```
Level: informational

Facility: LOCAL7

The server performed a readdir operation on the specified directory.

Removing file

```
Removing file '<filename>'
```
Level: notice

Facility: LOCAL7

The server is removing the specified file.

Failed to remove file

```
Failed to remove file '<filename>'. Error <error-number>.
```
Level: warning

Facility: LOCAL7

The server failed to remove the specified file.

Creating directory

```
Creating directory '<filename>'
```
Level: notice

Facility: LOCAL7

The server is creating the requested directory.

Failed to create directory

```
Failed to make dir '<filename>'. Error <error-number>.
```
Level: warning

Facility: LOCAL7

The server failed to create the requested directory.

Removing directory

```
Removing directory '<filename>'
```
Level: notice

Facility: LOCAL7

The server is removing the specified directory.

Failed to remove directory

```
Failed to remove dir '<filename>'. Error <error-number>.
```
Level: warning

Facility: LOCAL7

The server failed to remove the requested directory.

Resolving path

```
Resolving path to '<filename>'
```
Level: informational

Facility: LOCAL7

The server is resolving the path to the specified file.

Renaming file

```
Renaming file '<filename>' to '<new-filename>'
```
Level: notice

Facility: LOCAL7

The server is renaming the specified file.

Failed to rename file

```
Failed to rename '<filename>'. Error <error-number>.
```
Level: warning

Facility: LOCAL7

The server failed to rename the specified file.

Reading link

`Reading link '<link-name>'`
Level: informational

Facility: LOCAL7

The server is reading the specified link.

Readlink failed

`Readlink failed for '<link-name>'. Error <error-number>.`
Level: warning

Facility: LOCAL7

The readlink operation failed in the server.

Creating link

`Creating link '<link-name>' with value '<target>'`
Level: notice

Facility: LOCAL7

The server is creating the specified link.

Creating link failed

`Failed to create symlink '<link-name>'. Error <error-number>.`
Level: warning

Facility: LOCAL7

Creating the specified link failed in the server.

Got EXTENDED request

`Got EXTENDED request <request-number>`
Level: informational

Facility: LOCAL7

The server received an extended request from the client.

Appendix G Audit Messages

This appendix lists the audit messages generated by the server.

2000 Sft_server_starting
Level: notice
Origin: Tectia Secure File Transfer

The SFT server is starting.

Default log facility: daemon

Argument	Description
Username	Username
Session-Id	Session ID

2001 Sft_server_stopping
Level: notice
Origin: Tectia Secure File Transfer

The SFT server is stopping.

Default log facility: daemon

Argument	Description
Username	Username
Session-Id	Session ID

2002 Sft_server_connected
Level: informational
Origin: Tectia Secure File Transfer

The connection to SFT server was accepted.

Default log facility: daemon

Argument	Description
Session-Id	Session ID

2003 Sft_server_disconnected
Level: informational
Origin: Tectia Secure File Transfer

The connection to SFT server was disconnected.

Default log facility: daemon

Argument	Description
Session-Id	Session ID

2004 Sft_server_fxp_request
Level: informational
Origin: Tectia Secure File Transfer

Invalid SFT protocol message was received.

Default log facility: daemon

Argument	Description
Request type	Request type
Request ID	Request ID
Text	Error message
Session-Id	Session ID

2005 Sft_server_fxp_unknown
Level: notice
Origin: Tectia Secure File Transfer

Unknown SFT protocol message was received.

Default log facility: daemon

Argument	Description
Request type	Request type
Request ID	Request id
Session-Id	Session ID

2006 Sft_server_fxp_version
Level: notice
Origin: Tectia Secure File Transfer

SFT protocol version was negotiated.

Default log facility: daemon

Argument	Description
Negotiated version	Negotiated protocol version

Argument	Description
Session-Id	Session ID

2007 Sft_server_open_file

Level: notice

Origin: Tectia Secure File Transfer

File OPEN was successful.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2008 Sft_server_open_file_failed

Level: warning

Origin: Tectia Secure File Transfer

File OPEN failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2010 Sft_server_read_file_failed

Level: informational

Origin: Tectia Secure File Transfer

File READ failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2012 Sft_server_write_file_failed

Level: informational

Origin: Tectia Secure File Transfer

File WRITE failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2013 Sft_server_close_file
Level: informational
Origin: Tectia Secure File Transfer

File CLOSE was successful.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2014 Sft_server_close_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File CLOSE failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2015 Sft_server_stat_file
Level: informational
Origin: Tectia Secure File Transfer

File STAT was successful.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2016 Sft_server_stat_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File STAT failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2017 Sft_server_lstat_file
Level: informational
Origin: Tectia Secure File Transfer

File LSTAT was successful.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2018 Sft_server_lstat_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File LSTAT failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2019 Sft_server_fstat_file
Level: informational
Origin: Tectia Secure File Transfer

File FSTAT was successful.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2020 Sft_server_fstat_file_failed
Level: informational
Origin: Tectia Secure File Transfer

File FSTAT failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2021 Sft_server_setstat_file
Level: notice
Origin: Tectia Secure File Transfer

File SETSTAT was successful.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2022 Sft_server_setstat_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File SETSTAT failed.

Default log facility: daemon

Argument	Description
File name	File name

Argument	Description
Text	Audit message
Session-Id	Session ID

2023 Sft_server_fsetstat_file

Level: notice
Origin: Tectia Secure File Transfer

File FSETSTAT was successful.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2024 Sft_server_fsetstat_file_failed

Level: warning
Origin: Tectia Secure File Transfer

File FSETSTAT failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2025 Sft_server_remove_file

Level: notice
Origin: Tectia Secure File Transfer

File REMOVE was successful.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2026 Sft_server_remove_file_failed

Level: warning

Origin: Tectia Secure File Transfer

File REMOVE failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2027 Sft_server_rename_file
Level: notice
Origin: Tectia Secure File Transfer

File was renamed.

Default log facility: daemon

Argument	Description
Old file name	Old file name
New file name	New file name
Text	Audit message
Session-Id	Session ID

2028 Sft_server_rename_file_failed
Level: warning
Origin: Tectia Secure File Transfer

File RENAME failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2029 Sft_server_symlink
Level: informational
Origin: Tectia Secure File Transfer

Symlink was created.

Default log facility: daemon

Argument	Description
Link name	Link name

Argument	Description
Target name	Target name
Text	Audit message
Session-Id	Session ID

2030 Sft_server_symlink_failed
Level: warning
Origin: Tectia Secure File Transfer

File/directory SYMLINK failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2031 Sft_server_readlink
Level: informational
Origin: Tectia Secure File Transfer

Symlink was resolved.

Default log facility: daemon

Argument	Description
Link name	Link name
Target name	Resolved target name
Text	Audit message
Session-Id	Session ID

2032 Sft_server_readlink_failed
Level: warning
Origin: Tectia Secure File Transfer

File/directory READLINK failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2033 Sft_server_realpath
Level: informational

Origin: Tectia Secure File Transfer

File name was resolved.

Default log facility: daemon

Argument	Description
File name	File name
Path	Resolved file name
Text	Audit message
Session-Id	Session ID

2034 Sft_server_realpath_failed
Level: warning
Origin: Tectia Secure File Transfer

File/directory REALPATH failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2035 Sft_server_digest
Level: informational
Origin: Tectia Secure File Transfer

SFT server calculated digest.

Default log facility: daemon

Argument	Description
File name	File name
Offset	File offset
Length	Length of the file data for digest calculation
Digest	Calculated digest
Session-Id	Session ID

2036 Sft_server_digest_failed
Level: warning
Origin: Tectia Secure File Transfer

File DIGEST failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2037 Sft_server_open_dir

Level: notice
Origin: Tectia Secure File Transfer

Directory OPEN was successful.

Default log facility: daemon

Argument	Description
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2038 Sft_server_open_dir_failed

Level: warning
Origin: Tectia Secure File Transfer

Directory OPEN failed.

Default log facility: daemon

Argument	Description
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2039 Sft_server_close_dir

Level: informational
Origin: Tectia Secure File Transfer

Directory CLOSE was successful.

Default log facility: daemon

Argument	Description
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2040 Sft_server_close_dir_failed
Level: warning
Origin: Tectia Secure File Transfer

Directory CLOSE failed.

Default log facility: daemon

Argument	Description
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2041 Sft_server_create_dir
Level: notice
Origin: Tectia Secure File Transfer

Directory CREATE was successful.

Default log facility: daemon

Argument	Description
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2042 Sft_server_create_dir_failed
Level: warning
Origin: Tectia Secure File Transfer

Directory CREATE failed.

Default log facility: daemon

Argument	Description
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2043 Sft_server_remove_dir
Level: notice
Origin: Tectia Secure File Transfer

Directory REMOVE was successful.

Default log facility: daemon

Argument	Description
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2044 Sft_server_remove_dir_failed

Level: warning
Origin: Tectia Secure File Transfer

Directory REMOVE failed.

Default log facility: daemon

Argument	Description
Directory name	Directory name
Text	Audit message
Session-Id	Session ID

2045 Sft_server_read_dir

Level: informational
Origin: Tectia Secure File Transfer

Directory READ was successful.

Default log facility: daemon

Argument	Description
Directory name	Directory name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2046 Sft_server_read_dir_failed

Level: warning
Origin: Tectia Secure File Transfer

Directory READ failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2047 Sft_server_stream_read

Level: informational
Origin: Tectia Secure File Transfer

File read stream was initiated.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Path	Path to socket file
Offset	File offset
Length	File transfer limit in bytes
Session-Id	Session ID

2048 Sft_server_stream_read_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM READ failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2049 Sft_server_stream_write
Level: informational
Origin: Tectia Secure File Transfer

File write stream was initiated.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Path	Path to socket file
Offset	File offset
Length	File transfer limit in bytes
Session-Id	Session ID

2050 Sft_server_stream_write_failed
Level: warning

Origin: Tectia Secure File Transfer

File STREAM WRITE failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2051 Sft_server_stream_cancel
Level: informational
Origin: Tectia Secure File Transfer

File STREAM CANCEL was successful.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2052 Sft_server_stream_cancel_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM CANCEL failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2053 Sft_server_stream_wait
Level: informational
Origin: Tectia Secure File Transfer

File STREAM WAIT was successful.

Default log facility: daemon

Argument	**Description**
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2054 Sft_server_stream_wait_failed

Level: warning

Origin: Tectia Secure File Transfer

File STREAM WAIT failed.

Default log facility: daemon

Argument	**Description**
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2055 Sft_server_download_begin

Level: notice

Origin: Tectia Secure File Transfer

File download begins.

Default log facility: daemon

Argument	**Description**
File name	File name
Limit	File transfer limit in bytes
Session-Id	Session ID

2056 Sft_server_download_end

Level: notice

Origin: Tectia Secure File Transfer

File download ended.

Default log facility: daemon

Argument	**Description**
File name	File name
Read	Number of bytes read
Success | Error	Error code
Session-Id	Session ID

2057 Sft_server_upload_begin

Level: notice

Origin: Tectia Secure File Transfer

File upload begins.

Default log facility: daemon

Argument	Description
File name	File name
Limit	File transfer limit in bytes
Session-Id	Session ID

2058 Sft_server_upload_end

Level: notice

Origin: Tectia Secure File Transfer

File upload ended.

Default log facility: daemon

Argument	Description
File name	File name
Wrote	Number of bytes written
Success | Error	Error code
Session-Id	Session ID

2059 Sft_server_statistics

Level: notice

Origin: Tectia Secure File Transfer

File statistics.

Default log facility: daemon

Argument	Description
File name	File name
Read	Number of bytes read
Wrote	Number of bytes written
Text	Audit message
Session-Id	Session ID

2060 Sft_server_operation_not_allowed

Level: notice

Origin: Tectia Secure File Transfer

Extended SFT protocol message was received, but the operation was prohibited.

Default log facility: daemon

Argument	Description
Request type	Request type
Request ID	Request ID
Ext request type	Extended request type
Text	Error message
Session-Id	Session ID

2061 Sft_server_operation_no_license
Level: notice
Origin: Tectia Secure File Transfer

Extended SFT protocol message was received, but no suitable license was found for the operation.

Default log facility: daemon

Argument	Description
Request type	Request type
Request ID	Request ID
Ext request type	Extended request type
Text	Error message
Session-Id	Session ID

2062 Sft_server_no_virtual_folder_dir
Level: error
Origin: Tectia Secure File Transfer

Virtual folder directory does not exist.

Default log facility: daemon

Argument	Description
Text	Virtual folder
Directory name	Directory name
Text	Reason

2065 Sft_server_stat_extended
Level: informational
Origin: Tectia Secure File Transfer

File STREAM STAT was successful.

Default log facility: daemon

Argument	Description
File name	File name

Argument	Description
Text	Audit message
Session-Id	Session ID

2066 Sft_server_stat_extended_failed
Level: informational
Origin: Tectia Secure File Transfer

File STREAM STAT failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2067 Sft_server_lstat_extended
Level: informational
Origin: Tectia Secure File Transfer

File STREAM LSTAT was successful.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2068 Sft_server_lstat_extended_failed
Level: informational
Origin: Tectia Secure File Transfer

File STREAM LSTAT failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2069 Sft_server_setstat_extended
Level: notice
Origin: Tectia Secure File Transfer

File STREAM SETSTAT was successful.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2070 Sft_server_setstat_extended_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM SETSTAT failed.

Default log facility: daemon

Argument	Description
File name	File name
Text	Audit message
Session-Id	Session ID

2071 Sft_server_stream_offset
Level: notice
Origin: Tectia Secure File Transfer

File STREAM OFFSET was successful.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle
Text	Audit message
Session-Id	Session ID

2072 Sft_server_stream_offset_failed
Level: warning
Origin: Tectia Secure File Transfer

File STREAM OFFSET failed.

Default log facility: daemon

Argument	Description
File name	File name
File handle	File handle

Argument	Description
Text	Audit message
Session-Id	Session ID

Appendix H Open Source Software License Acknowledgements

SSH Communications Security Corporation acknowledges the following Open Source Software used in the Tectia client/server solution.

BSD Software
> This product includes software developed by the University of California, Berkeley and its contributors.

DES
> This product includes software developed by Eric Young `eay@cryptsoft.com`.

ICU
> Copyright © 1995-2016 International Business Machines Corporation and others

> All rights reserved.

> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

OpenSSL

Copyright © 1998-2016 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. 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.

3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)"

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED 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 OpenSSL PROJECT OR ITS 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.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

PCRE

This software includes PCRE library. Copyright © 1997-2009 University of Cambridge. All rights reserved.

C++ wrapper functions. Copyright © 2007-2009, 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. zEDC zlib is provided by IBM.

Appendix I Setting ICSF Permissions in RACF for Cryptographic Hardware Support

To enable cryptographic hardware you need to enable the following CSFSERV profiles for all client, and server IDs in RACF. You can run the first set of commands with the Tectia SSH Assistant ISPF application job `1.5 CSFSERV`.

```
SETROPTS CLASSACT(CSFSERV CSFKEYS XCSFKEY)
SETROPTS RACLIST(CSFSERV) GENERIC(CSFSERV)
RDEFINE CSFSERV CSFRNG UACC(NONE)
PERMIT CSFRNG CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFIQA UACC(NONE)
PERMIT CSFIQA  CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFIQF UACC(NONE)
PERMIT CSFIQF  CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1TRC UACC(NONE)
PERMIT CSF1TRC CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1TRD UACC(NONE)
PERMIT CSF1TRD CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1SKE UACC(NONE)
PERMIT CSF1SKE CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1SKD UACC(NONE)
PERMIT CSF1SKD CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFOWH  UACC(NONE)
PERMIT CSFOWH  CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFCKM  UACC(NONE)
PERMIT CSFCKM CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFKPI2  UACC(NONE)
PERMIT CSFKPI2 CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFENC  UACC(NONE)
PERMIT CSFENC CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFDEC  UACC(NONE)
PERMIT CSFDEC CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFSAD  UACC(NONE)
PERMIT CSFSAD CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFSAE  UACC(NONE)
```

```
PERMIT CSFSAE CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSFHMG  UACC(NONE)
PERMIT CSFHMG CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1GAV  UACC(NONE)
PERMIT CSF1GAV CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1DVK  UACC(NONE)
PERMIT CSF1DVK CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1GKP  UACC(NONE)
PERMIT CSF1GKP CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1PKS  UACC(NONE)
PERMIT CSF1PKS CLASS(CSFSERV) ID(*) ACCESS(READ)
RDEFINE CSFSERV CSF1PKV  UACC(NONE)
PERMIT CSF1PKV CLASS(CSFSERV) ID(*) ACCESS(READ)
SETROPTS RACLIST(CSFSERV) REFRESH
```

If possible, avoid defining the following SAF/RACF profile. Otherwise you must grant READ access to this profile for all client and server IDs:

```
CLASS(CRYPTOZ) CLEARKEY.SYSTOK-SESSION-ONLY
```

To enable use of IBM Crypto Express Card (CEX) you also need to enable the following CSFSERV profiles for all client, and server IDs in RACF:

```
# For Cipher offload to CEX
RDEFINE CSFSERV CSFCKM  UACC(NONE)
PERMIT CSFCKM CLASS(CSFSERV) ID(*) ACCESS(READ)

RDEFINE CSFSERV CSFSAD  UACC(NONE)
PERMIT CSFSAD CLASS(CSFSERV) ID(*) ACCESS(READ)

RDEFINE CSFSERV CSFSAE  UACC(NONE)
PERMIT CSFSAE CLASS(CSFSERV) ID(*) ACCESS(READ)

# For MAC offload to CEX
RDEFINE CSFSERV CSFKPI2  UACC(NONE)
PERMIT CSFKPI2 CLASS(CSFSERV) ID(*) ACCESS(READ)

RDEFINE CSFSERV CSFHMG  UACC(NONE)
PERMIT CSFHMG CLASS(CSFSERV) ID(*) ACCESS(READ)
SETROPTS CLASSACT(CSFSERV)
SETROPTS RACLIST(CSFSERV) REFRESH
```

Appendix J Offloading SSHD2 with zIIP

Shifts mainframe z CPACF instructions to zIIP processor, if the processor is available. CPACF instructions are used for `3DES-CBC, AES-CBC, AES-CTR, AES-GCM, HMAC-SHA` algorithms.

Perform the following steps in order to enable zIIP processing:

Enable sshd2 program with APF authorized flag by using the command: `extattr +a sshd2` Alternatively do the following modifications in SSZ Assistant installer job `X09LOAD` changing `PGM /bin/extattr +p` into `PGM /bin/extattr +ap`

```
//* Set program-control attribute for server binary
//    IF  RC = 0 THEN
//PCTL    EXEC SSZPBPX
//STDPARM  DD  *
PGM /bin/extattr +ap
 /u/vendor/data0/opt/tectia/sbin/sshd2
//* CHANGE THE ABOVE PGM STATEMENT TO PGM /bin/extattr +ap
//*   TO ADD APF attribute for server binary
//    ENDIF
//*
//* log results of previous job steps
//LOG     EXEC PGM=IRXJCL,COND=EVEN,PARM='JOBLOGRX'
//SYSEXEC  DD  DSN=EXAMPLE.SSZ.INSTALL.CNTL,
//             DISP=SHR
//SYSTSPRT DD  DSN=EXAMPLE.SSZ.INSTALL.LOG,
//             DISP=MOD
//SYSTSIN  DD  DUMMY
//*
```

(This change can be reverted by specifying `PGM /bin/extattr -a`)

Update PGM parameter of EXEC statement in SSHD2 started procedure JCL to use BPXBATA8 or follow instructions in sample SSHD2 JCL in SSZ assistant tailored sample DSName (option 0.1). If updating from a previous version be sure to run SSZ Assistant installer job `X13PROC`

*Note!*If the BPXPRG setting is not matched with APF attribute of sshd2, the start of SSHD2 proc will fail.

```
//SSHD2   PROC OPTS=''
//*       SET  BPXPRG=BPXBATSL
//* UNCOMMENT BELOW STATEMENT IF sshd2 PROGRAM IS APF AUTHORISED
//        SET  BPXPRG=BPXBATA8
//TECTIA  EXEC PGM=&BPXPRG,REGION=0M,TIME=NOLIMIT,
//             PARM=('PGM /opt/tectia/sbin/sshd2 -F &OPTS')
//STDENV   DD  DSN=EXAMPLE.VXXXX.PARMLIB(SSHENV),DISP=SHR
//STDOUT   DD  SYSOUT=*
//*TDERR   DD  SYSOUT=*
//STDIN    DD  DUMMY
//        PEND
```

(To revert these changes, you only need to change comment `PGM=BPXBATA8` and uncomment `PGM=BPXBATSL`)

Index

Symbols

A

B

C

D