



SSH Tectia Server 6.0 for IBM z/OS Quick Start Guide

9 January 2009

This document gives instructions on getting started with SSH Tectia Server for IBM z/OS on IBM mainframes.

© 2005 - 2008 SSH Communications Security Corp.

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

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

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

SSH Communications Security Corp.
Valimotie 17, FIN-00380 Helsinki; Finland

<http://www.ssh.com/>

Contents

1	About This Document	5
1.1	Introduction to SSH Tectia Server for IBM z/OS	5
1.2	System Authorization Facility	6
1.3	Sample Files	6
1.4	Documentation Conventions	7
1.5	Customer Support	7
2	Installing SSH Tectia Server for IBM z/OS	9
2.1	Preparing for Installation	9
2.1.1	System Requirements	9
2.1.2	Permission Requirements	10
2.1.3	Directories and Datasets	10
2.1.4	Upgrading Previously Installed Secure Shell Software	12
2.2	Installing the SSH Tectia Server for IBM z/OS Software	13
2.2.1	Unpacking the Installation Package	14
2.2.2	Creating the SAMPLIB and PARMLIB Datasets	15
2.2.3	Preparing the System	15
2.2.4	Creating the SSHD2 User	16
2.2.5	Creating the /opt/tectia Directory	16
2.2.6	Running the Setup Script	18

3	Getting Started with SSH Tectia Server for IBM z/OS	19
3.1	Running the Server	19
3.2	Environment Variables for Server and Client Applications	20
3.3	Running Client Programs	21
3.3.1	Under USS	21
3.3.2	Under MVS	22
4	Setting up Non-Interactive Server and User Authentication	23
4.1	Key Distribution Tool	23
4.2	Authenticating Remote Server Hosts	24
4.2.1	Fetching Remote Server Keys	24
4.3	Using Password for User Authentication	26
4.4	Using Public Key for User Authentication	26
4.4.1	Distributing Mainframe User Keys	27
5	Setting up Non-Interactive Secure File Transfer	31
5.1	Controlling File Transfer	31
5.1.1	Enabling Example File Transfer Profiles	32
5.2	File Transfer Examples	32
5.2.1	File Transfer Example Using <code>scp3</code>	32
5.2.2	File Transfers Using REXX Scripts and a JCL Procedure	33

Chapter 1

About This Document

This document contains instructions on getting started with SSH 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 SSH Tectia Server for IBM z/OS
- getting started with SSH Tectia Server for IBM z/OS
- example of setting up non-interactive server and user authentication
- example of non-interactive file transfer

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

SSH Tectia Server for IBM z/OS Administrator Manual gives detailed instructions on the installation, configuration, and use of SSH Tectia Server for IBM z/OS on IBM mainframes.

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

1.1 Introduction to SSH Tectia Server for IBM z/OS

SSH 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 SSH Tectia Client and ConnectSecure on Windows, it provides transparent secure TN3270 connections.

The server component of SSH Tectia Server 6.0 for IBM z/OS is based on SSH Tectia Server 4.3 but it has the same advanced file transfer features as SSH Tectia Server 6.0 on Unix platforms.

The client components of SSH Tectia Server 6.0 for IBM z/OS are based on SSH Tectia Client 6.0 and they support the same advanced file transfer features as SSH Tectia ConnectSecure 6.0 on Unix platforms.

SSH Tectia Server 6.0 for IBM z/OS runs on z/OS versions 1.6 through 1.10. 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

SSH 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, SSH Tectia Server for IBM z/OS includes a directory called `/opt/tectia/doc/zOS/samples` that contains USS scripts, and a directory called `/opt/tectia/doc/zOS/SAMPLIB` that contains JCL scripts and other files that complement this document.

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.
- In JCL scripts, change the `STDENV` paths to point to the correct location on your system.
- In JCL scripts, make sure that the `STDOUT` and `STDERR` paths are unique so that two JCL scripts run concurrently never use the same paths.
- In (non-transparent) FTP tunneling examples, make sure that the local port number used for the secure listener is unique for each instance which is run concurrently.

Many of the tasks described by the samples can be accomplished by other means, for example by using TSO commands or OMVS Shell.

1.4 Documentation Conventions

The following special conventions are used in this document:

Convention	Usage	Example
Bold	Menus, GUI elements, strong emphasis	Click Apply or OK .
→	Series of menu selections	Select File → Save .
Monospace	Filenames, commands, directories, URLs etc.	Refer to <code>readme.txt</code> .
<i>Italics</i>	Reference to other documents or products, emphasis	See <i>SSH Tectia Client User Manual</i> .

Note: In the example commands in this document, the backslash character (\) at the end of a line denotes that the command should be entered on a single line (without the backslash character).

1.5 Customer Support

Please see *SSH Tectia Server for IBM z/OS Administrator Manual* for detailed instructions on installing, configuring, and using SSH Tectia Server for IBM z/OS. All SSH Tectia product documentation is available at <http://www.ssh.com/resources/>.

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

If you have purchased a maintenance agreement, you are entitled to technical support from SSH Communications Security. Review your agreement for specific terms.

Please see the following page for more information on submitting support requests, feature requests, or bug reports, and on accessing the available online resources: <http://www.ssh.com/support/contact/>.

Chapter 2

Installing SSH Tectia Server for IBM z/OS

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

2.1 Preparing for Installation

This section lists the necessary pre-requisites for installing SSH Tectia Server for IBM z/OS.

2.1.1 System Requirements

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

- z/OS 1.6, 1.7, 1.8, 1.9, and 1.10

The following minimum hardware is required:

- 1 GB RAM for hundreds of simultaneous tunnels
- Minimum 200 MB free disk space, 250 cylinders (includes client components)
During installation, an additional 300 MB of temporary disk space is required.
- CD-ROM drive on a PC with a network connection to mainframe
- TCP/IP connection

2.1.2 Permission Requirements

The following permissions are required for installing and running SSH Tectia Server for IBM z/OS:

File system requirements

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

User account requirements for installing the server

- The `setup` script 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.
- The user account running the setup must have an OMVS segment and the UID 0.

User account requirements for running the server

It is recommended that a user account, `SSHD2`, is created for running SSH Tectia Server for IBM z/OS, see Section 2.2.4 (Creating the `SSHD2` User):

- 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.

User account requirements for using SSH Tectia client programs

- Required: An OMVS segment
- Optional: A home directory, needed only for storing host keys or public keys

Library requirements

- `CEE.SCEERUN` and `CEE.SCEERUN2` libraries must be available in `LPALIB` or `LNKLST`.
- `CEE.SCEERUN2` must be program-controlled.

TCP permissions

The server must be allowed to listen to port 22 (or other configured Secure Shell port).

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.

2.1.3 Directories and Datasets

USS

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

/opt/tectia

Contains executable binaries, setup scripts, configuration files, server key files, manual pages, documentation, license agreement, example JCL scripts.

Space: 200 MB, 250 Cyls, read/write

After running the setup script, SSH Tectia Server for IBM z/OS consists of a directory structure under /opt/tectia. You may create a symbolic 'tectia' link to the place where the directory structure resides under /opt or you may want to create a separate HFS or zFS file system for it.

\$HOME/.ssh2

Each z/OS user account that runs SSH client programs or accesses the server must have a USS home directory (\$HOME, for example, /u/home1/username). The system will create the .ssh2 directory under the home directory. It contains the user's configuration files and keys.

Space: 128 kB, read/write

The runtime programs of SSH 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. Used also temporarily during installation.

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(1)` 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 `rwxrwxrwx`, they must be adjusted with `chmod(1)`:

```
:> 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 SSH 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 `osshell`.

2.1.4 Upgrading Previously Installed Secure Shell Software

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

From OpenSSH

Before installing SSH Tectia Server 6.0 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 [2.2](#) (Installing the Software).

From SSH Tectia Server for IBM z/OS Version 5.x

The product structure and the installation directory of SSH Tectia Server for IBM z/OS has changed in version 6.0. If you have an earlier version of SSH Tectia Server for IBM z/OS installed, it can coexist with version 6.0.

Before proceeding with the installation as described in [2.2](#) (Installing the Software), consider the following issues.

When you run `setup.sh` for upgrading from version 5.x you must choose if you want to keep the old 5.x installation intact or if you want to uninstall all 5.x files. You must run `setup.sh` either with

```
./setup.sh --uninstall-old
```

or

```
./setup.sh --keep-old
```

If you choose to uninstall 5.x, please note that the directories `/etc/ssh2` and `/usr/lpp/ssh2` will get unconditionally removed after the upgrade procedure has successfully completed. Make sure that those directories do not contain any files you want to keep for reference (ad-hoc configuration backups, notes etc.) before running `setup.sh`.

If you choose to keep the 5.x installation, nothing from the old 5.x will be uninstalled by the `setup.sh` script. Please note that by default the two installed servers compete for TCP port 22 and thus cannot be run simultaneously. To avoid conflicts at the next IPL please check and update:

- Your started task JCL for the server
- Operator instructions for running the server

- Any other `sshd2` init procedures (`/etc/rc` for example)

When upgrading from 5.x to 6.x, the default files/directories under `/etc/ssh2/` are copied from the 5.x installation to your new installation at `/opt/tectia/` and they continue to be used with SSH Tectia Server 6.0 for IBM z/OS. See *SSH Tectia Server for IBM z/OS Administrator Manual* for details.

The existing `/etc/ssh2/ssh2.config` or `$HOME/.ssh2/ssh2.config` configuration files are not used by SSH Tectia Server 6.0 for IBM z/OS client components. You have to create new `opt/tectia/etc/ssh-broker-config.xml` and `$HOME/.ssh2/ssh-broker-config.xml` files.

For information on upgrading SSH Tectia client tools configurations from to version 5.x to 6.0, see *SSH Tectia Server for IBM z/OS Migration Guide*.

From SSH Tectia Server for IBM z/OS Version 6.0

SSH Tectia Server 6.0 for IBM z/OS can be upgraded simply by installing a newer maintenance version of the software on top of the older version. The already defined configuration will be preserved untouched. If you have both versions 5.x and version 6.x installed you must choose which installation the upgraded system will be based on.

To upgrade SSH Tectia Server for IBM z/OS:

1. Unpack the archive as described in Section 2.2.1 (Unpacking the Archive).
2. Run the setup script as described in Section 2.2.6 (Running the Setup Script).

The setup script will not restart the server automatically.

2.2 Installing the SSH Tectia Server for IBM z/OS Software

SSH Tectia Server for IBM z/OS is installed to the USS area using an appropriate USS shell, for example an OMVS shell or a Telnet session.

The installation consists of the following steps:

1. Unpack the installation package. See Section 2.2.1 (Unpacking).
2. Create the `SAMPLIB` and `PARMLIB` datasets. See Section 2.2.2 (Creating `SAMPLIB`).
3. Prepare the system. See Section 2.2.3 (Preparing).
4. Create the `SSHD2` user. See Section 2.2.4 (Creating `SSHD2`).

5. Create the `/opt/tectia` directory. See Section 2.2.5 (Creating `/opt/tectia`).
6. Run the setup script. See Section 2.2.6 (Running Setup).

2.2.1 Unpacking the Installation Package

The contents of the installation package are binary programs and EBCDIC text files. Use binary mode to copy the package and unpack it without codeset conversion.

Before copying and unpacking the installation package, make sure that the temporary directory has at least 300 MB of free space.

Online Package

If you are installing from a downloaded online package, do the following steps:

1. Unpack the online package `ssh-tectia-server-zos-6.0.4.-ibmzos_1_6-comm.tar` (where `` is the current build number of the package) to a suitable temporary location, for example `/tmp` in the USS file system, with the following command:

```
> cd /tmp
> tar xvf ssh-tectia-server-zos-6.0.4.<b>-ibmzos_1_6-comm.tar
```

This will create the directory `/tmp/ssh-tectia-server-zos-6.0.4-ibmzos_1_6-comm` that contains `ssh-tectia-server-zos-6.0.4.-ibmzos_1_6.tar.Z` package.

2. Unpack the `ssh-tectia-server-zos-6.0.4.-ibmzos_1_6.tar.Z` package with the following commands:

```
> cd ssh-tectia-server-zos-6.0.4-ibmzos_1_6-comm
> tar xvzf ssh-tectia-server-zos-6.0.4.<b>-ibmzos_1_6.tar.Z
```

This will create the directory:

```
/tmp/ssh-tectia-server-zos-6.0.4-ibmzos_1_6/ssh-tectia-server-zos-6.0.4.<b>
```

CD-ROM

If you are installing from CD-ROM, do the following steps:

1. Copy the `ssh-tectia-server-zos-6.0.4.-ibmzos_1_6.tar.Z` package to a suitable temporary directory, for example `/tmp` in the USS file system.

2. Unpack the `ssh-TECTIA-server-zos-6.0.4.-ibmzos_1_6.tar.Z` package to a temporary location, for example `/tmp` in the USS file system, with the following command:

```
> tar xvzf ssh-TECTIA-server-zos-6.0.4.<b>-ibmzos_1_6.tar.Z
```

This will create the directory `/tmp/ssh-TECTIA-server-zos-6.0.4.`, where `` is the current build number of the package.

2.2.2 Creating the SAMPLIB and PARMLIB Datasets

The unpacked product includes a directory, `doc/zOS/SAMPLIB`, that contains sample JCL steps and commands, which are referred to in the following installation steps. To create the `SAMPLIB` and `PARMLIB` datasets, execute the following commands (replace `USERID` with correct user ID):

```
> cd /tmp/ssh-TECTIA-server-zos-6.0.4.<b>/doc/zOS/SAMPLIB
> tso "ALLOC DSN('USERID.SSZ.SRVR604.SAMPLIB') LRECL(80) RECFM(F, B) \
SPACE(2, 1) TRACKS DIR(5)"
> tso "ALLOC DSN('USERID.SSZ.SRVR604.PARMLIB') LRECL(80) RECFM(F, B) \
SPACE(2, 1) TRACKS DIR(5)"
> cp * "//'USERID.SSZ.SRVR604.SAMPLIB' "
> cp SSHENV "//'USERID.SSZ.SRVR604.PARMLIB' "
```

`SSHENV` is a parameter file that contains environment variables that are required for SSH Tectia binaries to run properly.

2.2.3 Preparing the System

SSH Tectia Server for IBM z/OS uses the ASCII and XPLINK features of the IBM Language Environment (LE) product. The libraries `CEE.SCEERUN` and `CEE.SCEERUN2` must be available in `LPALIB` or `LNKLST`.

The server is program-controlled. All the programs it loads must be program-controlled too, otherwise a `BPX015I` error will occur when the server tries to create a user process. To make `CEE.SCEERUN2` program-controlled, run the job `RACFPC` from `SAMPLIB` (shown below).

RACFPC:

```
//ADDUSR EXEC PGM=IKJEFT1A,DYNAMNBR=75,TIME=1440,REGION=6M
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTEM DD DUMMY
//SYSTSIN DD *
RALTER PROGRAM ** +
```

```

    ADDMEM('CEE.SCEERUN2'//NOPADCHK) +
    UACC(READ)
    SETROPTS WHEN(PROGRAM) REFRESH

```

If any crypto hardware devices are to be used, the machine or the LPAR must be enabled for cryptography. Depending on the machine and devices, the required feature may be Feature code 0800 or Feature code 3863.

2.2.4 Creating the SSHD2 User

The SSHD2 user is used to run the SSH Tectia Server and Certificate Validator. SSHD2 must have an OMVS segment and the UID 0. Further, if the BPX.DAEMON FACILITY class profile is defined, the user must have read access to it.

Caution: The SSHD2 user must not have SURROGATE rights. With those rights, the user could log in using an empty password, which is not desirable.

To create the SSHD2 user, use commands such as those in the ADDSSHD2 example located in the doc/zOS/SAMPLIB directory (shown below).

ADDSSHD2:

```

//ADDSSHD2 EXEC PGM=IKJEFT1A,DYNAMNBR=75,TIME=100,REGION=6M
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTEM DD DUMMY
//SYSTSIN DD *
  ADDUSER SSHD2 +
    NAME('User SSHD2 for running SSH Tectia server') +
    OWNER(IBMUSER) +
    NOPASSWORD NOOIDCARD +
    OMVS(PROGRAM('/bin/false') UID(0))
    PERMIT BPX.DAEMON CLASS(FACILITY) ID(SSHD2) ACCESS(READ)
    SETROPTS REFRESH RACLIST(FACILITY)
/*

```

2.2.5 Creating the /opt/tectia Directory

SSH Tectia Server for IBM z/OS is always installed to /opt/tectia.

To put the product in a separate HFS or zFS file system, make an empty tectia directory under /opt:

```

> cd /opt
> mkdir tectia

```

Create the file system and mount it on `/opt/tectia`. You can use the `CREAHFS` and `MOUNHFS` examples from `SAMPLIB` (shown below). The sample files are for HFS but zFS can also be used. Note that mounting the file system causes an operator query that must be responded to. You may want to copy the mount command into the `BPXPRMxx` member in `PARMLIB` so that the file system is mounted each time you IPL the system.

CREAHFS:

```
//CREAHFS EXEC PGM=IEFBR14
//*
//* Allocate a dataset for a USS file system. The space allocated, 250
//* cylinders, corresponds to about 200 MB on a 3390 device.
//*
//HFS      DD  DSN=SSZ.SRVR604.HFS,
//          UNIT=SYSALLDA,
//          SPACE=(CYL,(250,1,5)),
//          DSNTYPE=HFS,
//          DCB=(DSORG=PO),
//          DISP=(NEW,CATLG,DELETE)
```

MOUNHFS:

```
//* REPLY nn,dddd ON THE MACHINE CONSOLE TO ALLOW MOUNT
//*
//MOUNT    EXEC PGM=IKJEFT1A,DYNAMNBR=75,TIME=100,REGION=6M
//*
//* Mount the file system on an existing, empty directory
//*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTEM   DD DUMMY
//SYSTSIN  DD *
    MOUNT FILESYSTEM('SSZ.SRVR604.HFS') +
        MOUNTPPOINT('/opt/tectia') +
        TYPE(HFS) +
        MODE(RDWR)
/*
```

Setting Permissions to `/opt/tectia`

Ensure that the `/opt/tectia` directory is writable and that it is owned by the `SSHD2` user:

```
> chmod 755 /opt/tectia
> chown SSHD2:0 /opt/tectia
```

2.2.6 Running the Setup Script

Note: Before continuing the installation, make sure that all the required permissions, SSHD2 user account and `/opt/tectia/` directory are set. The user account used for running the setup script must have read access to the `BPX.FILEATTR.PROGCTL FACILITY`.

Note: If you are upgrading from an earlier version of SSH Tectia Server for IBM z/OS, read the instructions in Section 2.1.4 (Upgrading Previously Installed Secure Shell Software) before continuing.

Go to the directory into which the installation package was unpacked and run the setup script, for example:

```
> cd /tmp/ssh-tectia-server-zos-6.0.4.<b>  
> ./setup.sh
```

If you run the script in the TSO OMVS shell you may need to press the PF10 key from time to time to keep the shell in the RUNNING state.

The setup script does the following:

- It copies the binaries, manual pages, and configuration files under `/opt/tectia`.
- It makes the server program-controlled.
- It creates a host key pair (1536-bit RSA) if none is previously available at `/opt/tectia/etc/hostkey` and `/opt/tectia/etc/hostkey.pub`. For example, if the software is upgraded from an earlier version, a new key pair is not created but the existing key pair is used.

Chapter 3

Getting Started with SSH Tectia Server for IBM z/OS

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

3.1 Running the Server

To run `sshd2` as a started task, use a JCL procedure such as `SSHD2` from `SAMPLIB` (shown below). The JCL must be installed in the procedure library.

SSHD2:

```
//RUSSHD2 PROC F=START
//SSHD2 EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//          PARM='PGM /bin/sh /opt/tectia/etc/init.d/sshd2
//          &F foreground'
//STDOUT DD PATH='/tmp/SSHD2-sshd2.out',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDERR DD PATH='/tmp/SSHD2-sshd2.err',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDIN DD DUMMY
//          PEND
```

Start the server with the following operator command:

```
== > s sshd2
```

The `sshd2` job starts.

In the sample `SSHD2` script above, `sshd2` is started with the `foreground` option that disables the daemon mode. With the `foreground` option, the server does not spawn the process to background and the server task name stays as `sshd2`.

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

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

You can assign the user `SSHD2` 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
```

3.2 Environment Variables for Server and Client Applications

The environment variables `_CEE_RUNOPTS`, `_BPXK_AUTOCVT`, `_BPX_SHAREAS`, and `_BPX_BATCH_UMASK` must be set as shown in `SSHENV` and `sshsetenv` when running SSH Tectia Server for IBM z/OS programs (see below). The server startup procedures described in Section 3.1 (Running the Server) set these variables automatically for the server.

The `NAME=VALUE` format (as in `SSHENV`) is used when the client or server programs are run under MVS, and the Bourne shell format (as in `sshsetenv`) is used when the programs are run from a USS command line.

Note: The environment files must not contain line numbers or reading them will fail.

SSHENV:

```
_CEE_RUNOPTS=FILETAG(AUTOCVT,NOAUTOTAG),TRAP(ON)
_BPXK_AUTOCVT=ON
_BPX_SHAREAS=NO
_BPX_BATCH_UMASK=0022
SSH_BATCHMODE=ON
SSH_DEBUG_FMT="%W(72)(2) %Dd/%Dt/%Dy %Dh:%Dm:%Ds:%Df %m/%s:%n:%f %M"
```

sshsetenv:

```
_CEE_RUNOPTS="FILETAG(AUTOCVT,NOAUTOTAG),TRAP(ON)"
export _CEE_RUNOPTS
_BPXX_AUTOCVT=ON
export _BPXX_AUTOCVT
_BPX_BATCH_UMASK=0022
export _BPX_BATCH_UMASK
_BPX_SHAREAS=NO
export _BPX_SHAREAS
SSH_DEBUG_FMT="%W(72)(2) %Dd/%Dt/%Dy %Dh:%Dm:%Ds:%Df %m/%s:%n:%f %M"
export SSH_DEBUG_FMT
```

3.3 Running Client Programs

SSH Tectia Server for IBM z/OS contains three client-side applications:

- `sshg3` is a secure replacement for Telnet and other unsecured terminal applications. `sshg3` can also be used for remote command and job execution, and creating secure tunnels for TCP applications.
- `scpg3` is a secure replacement for remote copy (`rcp`) and provides easy secure non-interactive file transfers.
- `sftpg3` is a secure replacement for FTP and provides a user interface for interactive file transfers and a batch mode for unattended file transfers.

The `ssh-broker-g3` component handles all cryptographic operations and authentication-related tasks for the `sshg3`, `scpg3`, and `sftpg3` client programs. Normally, the Connection Broker starts in the background automatically whenever one of the SSH Tectia client programs is started, and stops when the client program is stopped.

The `ssh-socks-proxy` component is used for transparent FTP tunneling and FTP-SFTP conversion. For more information on the SSH Tectia SOCKS Proxy, see *SSH Tectia Server for IBM z/OS Administrator Manual*.

3.3.1 Under USS

Interactive remote sessions and file transfers can be used from Unix System Services shells. For example, OMVS, Telnet, or Secure Shell sessions can be used. **If OMVS shell is used, only non-interactive authentication methods can be used.**

For information on the command syntax and options, see the `sshg3`, `scpg3`, and `sftpg3` man pages.

For file transfer examples, please see Chapter 5 (Transferring Files).

3.3.2 Under MVS

SSH Tectia Server for IBM z/OS client-side applications can be executed in JCL by `BPXBATCH`, `BPXBATSL`, or `osshell`. `scp93` uses the same syntax for interactive and unattended file transfers. `sftp93` has a batch mode for non-interactive file transfers.

For easier user experience, file transfer client applications can also be run using a file transfer JCL procedure provided in `SAMPLIB`.

User interaction is not possible when using unattended file transfers. For unattended use, users must be set up to use a non-interactive authentication method, like a public key without a passphrase.

Because user interaction is not possible, the server host key must be stored on disk on the client before unattended file transfers will succeed. More information and examples on storing remote server keys can be found in Sections [4.2](#) (Authenticating Remote Server Hosts) and [4.2.1](#) (Fetching Remote Server Keys).

For unattended and JCL PROC file transfer examples, see Chapter [5](#) (Transferring Files).

Chapter 4

Setting up Non-Interactive Server and User Authentication

The Secure Shell protocol used by SSH 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.

The Secure Shell client users can authenticate themselves using password, public-key, keyboard-interactive, or host-based authentication. These authentication methods can be combined or used separately, depending on the level of functionality and security you want.

This chapter gives instructions for setting up non-interactive authentication for server and users using public keys. For information on the other authentication methods, see *SSH Tectia Server for IBM z/OS Administrator Manual*.

Most of the examples in this chapter are executed from Unix shell (for example, OMVS shell), but the same commands can also be run in JCL using BPXBATCH.

4.1 Key Distribution Tool

File transfer processing on mainframes is usually non-interactive. This means that the host keys of the remote servers must be stored in such a way that user interaction is not needed during the batch process, and that both users and processes use non-interactive authentication methods for user authentication.

The key distribution tool, `/opt/tectia/bin/ssh-keydist-g3`, can be used for storing multiple remote host keys to user-specific or common key store and setting up public-key authentication to multiple

hosts.

For detailed information on the command syntax of `ssh-keydist-g3`, see the man page.

4.2 Authenticating Remote Server Hosts

Remote Secure Shell servers are authenticated by a public-key procedure. 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 normally 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.

Caution: When `ssh-keydist-g3` is run with the `-N` option, it accepts the received host keys automatically without prompting the user. You should verify the validity of keys after receiving them or you risk being subject to a man-in-the-middle attack.

When the host key is received during the first connection to a remote host (or when the host key has changed) and you choose to save the key, its filename is by default stored in hashed format, `keys_hhh...`, where `hhh` is a hash of the host port and name. The saved file contains a hash of the host's public key. The hashed host key format is a security feature to make address harvesting on the hosts difficult.

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

The key storage format can be set in the `ssh-broker-config.xml` configuration file, or on the `ssh-keydist-g3` command line with the `-F` option. The command-line option takes precedence over the setting in the configuration file.

4.2.1 Fetching Remote Server Keys

The SSH Tectia clients on the mainframe must have the remote server public keys or public key hash values available in order to authenticate the remote server they are connecting to. The keys or key hash values can be stored in the mainframe user's `$HOME/.ssh2/hostkeys` directory or in the `/opt/tektia/etc/hostkeys` directory which is common for all the users. The key distribution tool can be used to retrieve multiple remote host keys and store the keys or key hash values to the user's host key directory or to the system-wide key store that is available for all the users.

For more information about hashed host key format, see Section 4.2 (Authenticating Remote Server Hosts).

Examples of Fetching Remote Server Keys

The following examples illustrate using `ssh-keydist-g3` for fetching remote server host keys.

Caution: When `ssh-keydist-g3` is run with the `-N` option, it accepts the received host keys automatically without prompting the user. You should verify the validity of keys after receiving them or you risk being subject to a man-in-the-middle attack.

Example 1: Using USS This example is run under USS shell. Multiple host keys are fetched in verbose mode and saved in plain format under the user's `$HOME/.ssh2/hostkeys` directory. The host keys are also saved using the IP addresses of the hosts. The log is stored under `/tmp`. The log will list the accepted keys and their fingerprints. You should verify them after running the command.

```
> ssh-keydist-g3 --verbose --accept-host-keys --accept-host-keys-also-by-ip \
--accepted-host-key-filename-format plain \
--accepted-host-key-log /tmp/newhosts.log \
host1 host2 host3
```

Example 2: Using JCL This example `HOSTSAVE` from `SAMPLIB` presents a JCL script that does the same steps as the USS command in Example 1 above (options are given in short format):

```
//HOSTSAVE EXEC PGM=IKJEFT1A,
//          REGION=OM
//SYSTSPRT DD  SYSOUT=*
//STDOUT   DD  PATH='/tmp/&SYSUID.-HOSTSAVE.out',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDERR   DD  PATH='/tmp/&SYSUID.-HOSTSAVE.err',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDENV   DD  DSN=&SYSUID..SSZ.SRVR604.PARMLIB(SSHENV),
//          DISP=SHR
//SYSTSIN  DD  *
    BPXBATCH SH /opt/tectia/bin/ssh-keydist-g3 +
                -v -N -i -F plain -A /tmp/newhosts.log +
                host1 host2 host3
/*
//*
//PROUT    EXEC PGM=IKJEFT1A,
//          PARM='OCOPY INDD(STDOUT) OUTDD(STDOUTPR) TEXT'
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  DUMMY
//STDOUT   DD  PATH='/tmp/&SYSUID.-HOSTSAVE.out',
```

```

//          PATHOPTS=(ORDONLY),
//          PATHDISP=(DELETE,KEEP),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDOUTPR DD  SYSOUT=*,
//          DCB=(LRECL=4000,RECFM=VB)
//*
//PRERR     EXEC PGM=IKJEFT1A,
//          PARM='OCOPY INDD(STDERR) OUTDD(STDERRPR) TEXT'
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN   DD  DUMMY
//STDERR    DD  PATH='/tmp/&SYSUID.-HOSTSAVE.err',
//          PATHOPTS=(ORDONLY),
//          PATHDISP=(DELETE,KEEP),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDERRPR DD  SYSOUT=*,
//          DCB=(LRECL=4000,RECFM=VB)
//*

```

4.3 Using Password for User 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.

There are two cases where it cannot be used in SSH Tectia Server for IBM z/OS:

- When running SSH Tectia client programs from JCL there is no facility for getting the password interactively from the user. You can set up password in a file or dataset, or preferably use public-key authentication and a private key without a passphrase. See Section 4.4 (Using Public Key for User Authentication).
- Password authentication is not available when running SSH Tectia client programs from the TSO OMVS shell because the shell cannot guarantee that the password will be hidden on the screen. Use a Telnet shell or Secure Shell, or set up public-key authentication using a private key without a passphrase.

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

4.4 Using Public Key for User 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.

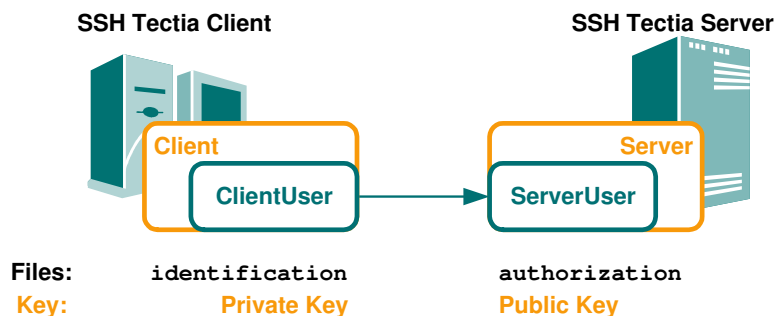


Figure 4.1: User public-key authentication

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

4.4.1 Distributing Mainframe User Keys

Administrators and other people can use passwords or public-key pairs with a passphrase-protected private key to access remote machines with SSH Tectia client tools from a Telnet or Secure Shell session. They can also use public-key pairs with a null passphrase if they want to run the SSH Tectia client programs in JCL.

Mainframe batch users are accounts that represent applications or subsystems, not people. They are set up with public-key pairs with a null passphrase to enable non-interactive access through JCL to remote servers. One key pair is generated for each batch user. If the batch user has a shared home directory, the key is placed in the shared `$HOME/.ssh2` directory, otherwise it is copied to the user's home directories on all the LPARs.

When the `ssh-keygen-g3` (or `ssh-keydist-g3`) tool is run with the `-P` option, which requests a null passphrase, it can be run from the OMVS shell or in JCL. It must be run under the batch user's user ID in order for the file permissions to be set properly.

The batch user accesses the remote machine using an account created and administered on the remote machine. The remote username may either be the same as the batch user's RACF user ID, or the same but in lower case, or a different username. Several batch users may use the same remote account. One batch user may use separate accounts on one remote machine for different accesses.

Each batch user's public key must be distributed to all the remote accounts it will be accessing. The way the public key is set up differs between SSH Tectia and OpenSSH. The `ssh-keydist-g3` script must be told which type of server the remote machine has. The server must be running when `ssh-keydist-g3` is run.

`ssh-keydist-g3` uses password authentication for this initial access to the remote server. The password for the remote account can be entered in a dataset or in a file. See Sections 4.4.1 (File) and 4.4.1 (Dataset) for instructions. The filename is entered as one of the options in the `ssh-keydist-g3` command.

The other options needed on the `ssh-keydist-g3` command line are the remote account username, the remote host DNS name or IP address, and the type of the remote Secure Shell server (SSH Tectia Server on Unix, SSH Tectia Server on Windows, SSH Tectia Server for IBM z/OS on mainframe, or OpenSSH on

Unix).

Password from File

To set up password-from-file authentication:

1. Create a file, for example `/home/userid/passwd_file`.
2. Make sure the file is readable only by the user that created it:


```
> chmod 600 /home/userid/passwd_file
```
3. Edit the file with your favorite text editor to contain one line with your password on the remote system, for example:

```
MyPasS
```

Password from Dataset

To set up password-from-dataset authentication:

1. Allocate a dataset or a dataset member, for example:


```
// 'USERID.PASSWD'
```
2. Make sure that the dataset is accessible only by the correct UserID.
3. Edit the password dataset to contain your password on the remote system. The format of the password dataset is one line containing only the password. For example:

```
MyPasS
```

Examples of Distributing User Keys

The following examples illustrate using `ssh-keydist-g3` for distributing user keys.

Example 1: Public-Key Upload to Unix OpenSSH Server from USS Shell This command creates a 1024-bit RSA key with an empty passphrase and uploads it to a Unix server running OpenSSH, including the necessary conversions. Public-key upload uses password-from-file for authentication. A log of the operation is stored under `/tmp`. The example assumes that the server host key has already been fetched and verified.

```
> ssh-keydist-g3 --key-type rsa --key-bits 1024 --empty-passphrase \
  --remote-user userid --password-file /home/userid/passwd_file \
  --user-key-log /tmp/my_log_file --openssh-unix open_server.example.com
```

Example 2: Public-Key Upload to Unix OpenSSH Server Using JCL This example KEYDIST from SAMPLIB presents a JCL script that does the same steps as the USS command in Example 1 above (options are given in short format):

```
//KEYDIST EXEC PGM=IKJEFT1A,
//          REGION=0M
//SYSTSPRT DD  SYSOUT=*
//STDOUT   DD  PATH='/tmp/&SYSUID.-KEYDIST.out',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDERR   DD  PATH='/tmp/&SYSUID.-KEYDIST.err',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDENV   DD  DSN=&SYSUID..SSZ.SRVR604.PARMLIB(SSHENV),
//          DISP=SHR
//SYSTSIN  DD  *
    BPXBATCH SH /opt/tectia/bin/ssh-keydist-g3 +
                -t rsa -b 1024 -P +
                -u userid -p "'/USERID.PASSWD'" +
                -U /tmp/my_log_file +
                -O host1.example.com
/*
//*
//PROUT    EXEC PGM=IKJEFT1A,
//          PARM='OCOPY INDD(STDOUT) OUTDD(STDOUTPR) TEXT'
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  DUMMY
//STDOUT   DD  PATH='/tmp/&SYSUID.-KEYDIST.out',
//          PATHOPTS=(ORDONLY),
//          PATHDISP=(DELETE,KEEP),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDOUTPR DD  SYSOUT=*,
//          DCB=(LRECL=4000,RECFM=VB)
//*
//*
//PRERR    EXEC PGM=IKJEFT1A,
//          PARM='OCOPY INDD(STDERR) OUTDD(STDERRPR) TEXT'
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  DUMMY
//STDERR   DD  PATH='/tmp/&SYSUID.-KEYDIST.err',
//          PATHOPTS=(ORDONLY),
//          PATHDISP=(DELETE,KEEP),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDERRPR DD  SYSOUT=*,
//          DCB=(LRECL=4000,RECFM=VB)
```

```
//*
```

Example 3: Public-Key Distribution to Multiple Hosts from USS Shell This example distributes an existing public key to several remote hosts automatically. Individual user names can be defined for each server. Server type (SSH Tectia Unix, SSH Tectia Windows, SSH Tectia z/OS, OpenSSH) needs to be defined with the flags: `-S`, `-W`, `-Z`, or `-O`. The example assumes that the relevant server host keys have already been fetched and verified.

In this example you can find four server "blocks":

- `-O -u user1 open_server.example.com`
- `-S -u user2 tectia_unix.example.com`
- `-W -u user2 tectia_win.example.com`
- `-Z -u user3 tectia_zos.example.com`

A password file is defined for each separate user ID. `user2` is assumed to have the same password on Unix and Windows. A log of the operation is stored under `/tmp`.

The command is as follows:

```
> ssh-keydist-g3 -f /home/userid/.ssh2/id_rsa_2048_a.pub \  
-U /tmp/userkeys.log \  
-p /home/userid/passwd_file1 \  
-O -u user1 open_server.example.com \  
-p /home/userid/passwd_file2 \  
-S -u user2 tectia_unix.example.com \  
-W -u user2 tectia_win.example.com \  
-p /home/userid/passwd_file3 \  
-Z -u user3 tectia_zos.example.com
```

Chapter 5

Setting up Non-Interactive Secure File Transfer

SSH Tectia Server for IBM z/OS provide security to existing FTP file transfers by applying the Secure File Transfer Protocol (SFTP) instead of FTP, or by using tunnels that encrypt the connection from the FTP client to the FTP server.

Unattended, automated file transfers between servers can be secured with the versatile command-line SFTP and SCP tools that apply the SFTP protocol.

This chapter contains examples of non-interactive secure file transfers with SSH Tectia Server for IBM z/OS. For more information on secure file transfer, see *SSH Tectia Server for IBM z/OS User Manual*.

5.1 Controlling File Transfer

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

SSH Tectia Server for IBM z/OS needs more information: which transfer format to use, what code sets are involved, and what the file characteristics are. SSH Tectia introduces some extensions to SFTP and the information can be relayed by using the Site commands of `scp3` and `sftp3`. Alternatively file transfer profiles can be used.

For command descriptions, see the `site` and `lsite` command on the `sftp3` man page and the `--dst-site` and `--src-site` options on the `scp3` man page.

A file transfer profile is a mechanism for pre-configuring different types of secure file transfers. Both the mainframe clients (`scp3`, `sftp3`) and the server use the same profile mechanism. There are two types of

profiles: named profiles and filename-matched profiles.

5.1.1 Enabling Example File Transfer Profiles

Example file transfer profiles can be enabled by copying the example profile file `/opt/tectia/etc/ssh_ftadv_config.example` to `/opt/tectia/etc/ssh_ftadv_config`. File transfer profiles for SSH Tectia Server for IBM z/OS can be set in files `/opt/tectia/etc/ssh_ftadv_config` (globally for all users) and `$HOME/.ssh2/ssh_ftadv_config` (for a specific user).

```
> cp /opt/tectia/etc/ssh_ftadv_config.example /opt/tectia/etc/ssh_ftadv_config
```

5.2 File Transfer Examples

5.2.1 File Transfer Example Using `scp3`

This example uses the SCPGET JCL that can be found from SAMPLIB. SAMPLIB contains also other `scp3` and `sftp3` non-interactive file transfer examples.

This example executes `scp3` and copies a remote file `textfile.txt` into a dataset `//'USERID.TEST.TEXTFILE'`. If the dataset does not exist, it is created with default values `recfm VB` and `lrecl 1024`.

```
//SCPGET EXEC PGM=IKJEFT1A,
//          DYNAMNBR=75,
//          TIME=1440,
//          REGION=6M
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSTEM    DD  DUMMY
//STDOUT    DD  PATH='/tmp/&SYSUID.-SCPGET.out',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDERR    DD  PATH='/tmp/&SYSUID.-SCPGET.err',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR)
//STDENV    DD  DSN=&SYSUID..SSZ.SRVR604.PARMLIB(SSHENV),
//          DISP=SHR
//SYSTSIN   DD  *
      BPXBATCH PGM /opt/tectia/bin/scp3 +
      user1@remote_host:textfile.txt +
      //'USERID.TEST.TEXTFILE'
```

```

/*
//STDPD    EXEC  PGM=IKJEFT1A,
//          DYNAMNBR=75,
//          TIME=1440,
//          REGION=6M
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSTEM   DD  DUMMY
//STDOUT   DD  PATH='/tmp/&SYSUID.-SCPGET.out',
//          PATHOPTS=(ORDONLY),
//          PATHDISP=(DELETE,KEEP)
//STDERR   DD  PATH='/tmp/&SYSUID.-SCPGET.err',
//          PATHOPTS=(ORDONLY),
//          PATHDISP=(DELETE,KEEP)
//STDOUTPR DD  SYSOUT=*,
//          DCB=(LRECL=4000,RECFM=VB)
//STDERRPR DD  SYSOUT=*,
//          DCB=(LRECL=4000,RECFM=VB)
//SYSTSIN  DD  *
    OCOPY INDD(STDOUT) OUTDD(STDOUTPR) TEXT
    OCOPY INDD(STDERR) OUTDD(STDERRPR) TEXT
/*

```

5.2.2 File Transfers Using REXX Scripts and a JCL Procedure

SSH Tectia Server for IBM z/OS contains example file transfer procedure and REXX functions for `scp3`, `sftp3`, and `ssh3` client applications.

The following PROC, REXX functions, and examples can be found from `SAMPLIB` and they can be easily modified according to the customer environment and needs. The REXX functions can also be called from user-written REXX programs.

- `SSZJSAMP`: Comprehensive JCL example file containing file transfer and remote command examples
- `SSZJSFTP`: Example JCL file for running multiple `sftp3` commands
- `SSZP`: JCL Procedure for running REXX functions

Enabling the File Transfer JCL Procedure

By default, the JCL procedure is set to run from the default `SAMPLIB` location, `//'&SYSUID..SSZ.SRVR604.SAMPLIB'`. If the procedure and REXX functions are run from some other location, modify the `EXECLIB` parameter on the `SSZP` procedure accordingly.

Example 1: SSZJSFTP In this example multiple `sftp3` file transfer commands are run using the SSZP procedure:

```
//SFTP EXEC SSZP,  
// PARM='SSZRFT'  
//*  
//SFTCMDS DD *  
open username@unix_server.example.com  
sget text_file.txt //'USERID.TEST.DATA.SET'  
sput //'USERID.TEST.DATA.SET' demo_file.txt  
lrm //TEST.DATA.SET  
sget demo_file.txt //PDS(MEM2)  
rm demo_file.txt  
sput //'USERID.PDS(MEM2)' member2  
ascii  
lsite O=FB R=80  
sget jcl.txt //'USERID.JCLLIB(JCL1)'
```

Index

- ADDSSHD2, [16](#)
- authentication, [23](#)
- authentication methods, [23](#)
- authentication: password, [26](#)
- authentication: public-key, [24](#), [26](#)

- contacting support, [7](#)
- CREAHFS, [17](#)
- creating SSHD2 user, [16](#)
- customer support, [7](#)

- disk space requirement, [9](#)
- distributing keys, [23](#)
- documentation, [5](#)
- documentation conventions, [7](#)

- environment variables, [20](#)

- file transfer examples, [32](#)
- file transfer profile, [31](#)

- getting support, [7](#)

- hashed host key format, [24](#)
- host key: hashed format, [24](#)

- installing SSH Tectia Server for IBM z/OS, [13](#)

- key distribution, [23](#)

- lsite, [31](#)

- MOUNHFS, [17](#)
- MVS, [22](#)

- non-interactive authentication, [23](#)
- non-interactive file transfer, [31](#)

- OMVS segment, [10](#)

- password authentication, [26](#)
- permission requirements, [10](#)
- public-key authentication: server, [24](#)
- public-key authentication: user, [26](#)

- RACFPC, [15](#)
- related documents, [5](#)
- removing old versions, [12](#)
- running client programs, [21](#)

- sample files, [6](#)
- samples, [6](#)
- SAMPLIB, [6](#)
- scp3, [21](#), [32](#)
- SCPGET, [32](#)
- server authentication methods, [23](#)
- sftp3, [21](#)
- site, [31](#)
- site commands, [31](#)
- ssh-keydist-g3, [23](#), [24](#), [27](#)
- ssh-keygen-g3, [26](#)
- SSHD2, [19](#)
- SSHENV, [20](#)
- sshg3, [21](#)
- sshsetenv, [20](#)
- support: contacting, [7](#)
- system requirements, [9](#)

- TCP permissions, [10](#)
- technical support, [7](#)

- upgrading SSH Tectia Server for IBM z/OS, [12](#)
- user authentication methods, [23](#)
- user authentication with password, [26](#)
- user authentication with public key, [26](#)
- user requirements, [10](#)
- USS, [21](#)