Tectia® Server 6.6 for IBM z/OS: User Manual

07 December 2020

Copyright © 2007–2020 SSH Communications Security Corporation

This software and documentation are protected by international copyright laws and treaties. All rights reserved.

ssh® and Tectia® are registered trademarks of SSH Communications Security Corporation in the United States and in certain other jurisdictions.

SSH and Tectia logos and names of products and services are trademarks of SSH Communications Security Corporation. Logos and names of products 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 Corporation.

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

For Open Source Software acknowledgements, see appendix Open Source Software License Acknowledgements in the Administrator Manual.

SSH Communications Security Corporation
Kornetintie 3, FI-00380 Helsinki, Finland
# Table of Contents

1. About This Document ........................................................................................................... 7  
   1.1. Documentation Conventions ............................................................................................ 7  
      1.1.1. Operating System Names .......................................................................................... 8  
   1.2. Customer Support .......................................................................................................... 9  
   1.3. Terminology ................................................................................................................. 9  

2. Getting Started .................................................................................................................... 13  
   2.1. Product Components ..................................................................................................... 13  
   2.2. Environment Variables for Client Applications ............................................................... 13  
   2.3. Running Client Programs ............................................................................................... 14  
      2.3.1. Under USS ........................................................................................................... 15  
      2.3.2. Under MVS .......................................................................................................... 15  
      2.3.3. Enabling Use of IBM Crypto Express Card (CEX) ....................................................... 16  
   2.4. Running the Connection Broker ...................................................................................... 16  
      2.4.1. Starting ssh-broker-g3 Manually under USS .............................................................. 17  
      2.4.2. Stopping ssh-broker-g3 .......................................................................................... 17  
      2.4.3. Reconfiguring ssh-broker-g3 .................................................................................. 17  
   2.5. Connecting to a Remote Host ......................................................................................... 17  
      2.5.1. Authenticating Remote Server Hosts ......................................................................... 17  
      2.5.2. Using Password Authentication .............................................................................. 18  
      2.5.3. Using Public-Key Authentication ............................................................................ 19  
      2.5.4. Logging in with Command-Line sshg3 ...................................................................... 19  

3. Configuring Client Tools ..................................................................................................... 21  
   3.1. Client Configuration Files .............................................................................................. 21  
      3.1.1. Editing the Configuration Files ................................................................................ 22  
   3.2. Environment Variables .................................................................................................. 22  
   3.3. Command-Line Options ............................................................................................... 22  

4. Authentication ..................................................................................................................... 23  
   4.1. Supported User Authentication Methods ........................................................................ 24  
   4.2. Server Authentication with Public Keys in File .............................................................. 24  
      4.2.1. Host Key Storage Formats ...................................................................................... 25  
      4.2.2. Using the System-Wide Host Key Storage ............................................................... 27
4.2.3. Resolving Hashed Host Keys ................................................................. 29
4.2.4. Using the OpenSSH known_hosts File .................................................. 30
4.3. Server Authentication with Certificates ..................................................... 31
  4.3.1. CA Certificates Stored in File .............................................................. 32
  4.3.2. CA Certificates Stored in SAF ............................................................. 33
  4.3.3. Server Certificates Stored in SAF ......................................................... 34
4.4. User Authentication with Passwords ......................................................... 35
  4.4.1. Password Stored in a File or Data Set .................................................... 36
4.5. User Authentication with Public Keys in a File .......................................... 37
  4.5.1. Creating Keys with ssh-keygen-g3 on z/OS ........................................... 38
  4.5.2. Uploading Public Keys from z/OS to Remote Host ............................... 39
  4.5.3. Using Keys Generated with OpenSSH ................................................. 42
4.6. User Authentication with Certificates ....................................................... 42
  4.6.1. Certificates Stored in File ................................................................. 43
  4.6.2. Certificates Stored in SAF ................................................................. 44
4.7. Host-Based User Authentication ............................................................... 45
4.8. User Authentication with Keyboard-Interactive ........................................ 46
4.9. Distributing Public Keys Using the Key Distribution Tool ......................... 46
  4.9.1. Fetching Remote Server Keys ............................................................. 47
  4.9.2. Distributing Mainframe User Keys .................................................... 49

5. System Administration .................................................................................. 53
  5.1. Defining Shell Access ............................................................................... 53
    5.1.1. Setting Codepage for Remote Account ............................................... 53
  5.2. Running Remote Commands ..................................................................... 54
    5.2.1. Remote Command Examples from USS ............................................. 54
    5.2.2. Remote Command Examples using JCL ........................................... 54
  5.3. Managing JCL Jobs over SFTP ................................................................. 55
    5.3.1. Tectia scpg3 ................................................................................. 55
    5.3.2. Tectia sftpg3 .............................................................................. 57
    5.3.3. OpenSSH sftp ............................................................................. 58
  5.4. Securing the Client .................................................................................. 59
    5.4.1. Disabling Agent Forwarding .......................................................... 59

6. Secure File Transfer Using SFTP .................................................................. 61
  6.1. Native z/OS FTP commands versus Tectia SFTP commands .................. 61
  6.2. Secure File Transfer with scpg3 and sftpg3 Commands ......................... 64
    6.2.1. Using scpg3 ................................................................. 64
    6.2.2. Using sftpg3 ................................................................. 65
  6.3. Handling MVS Data Sets and HFS File System Access ......................... 65
    6.3.1. Data Set and HFS File System Access ............................................... 66
    6.3.2. Data Set Access Using DD Cards ..................................................... 66
    6.3.3. Accessing Generation Data Groups (GDG) ........................................ 67
    6.3.4. Accessing Migrated Data Sets ....................................................... 68
Chapter 1 About This Document

This document describes using and configuring the Secure Shell client-side tools included in Tectia Server for IBM z/OS. It is meant for users and administrators of Tectia Server for IBM z/OS. The document also contains examples of using other Secure Shell client software to connect to a mainframe that runs Tectia Server for IBM z/OS.

This document contains the following information:

- Getting started
- Configuring Tectia client tools for z/OS
- Authentication
- System administration
- Transferring files using SFTP
- Tunneling
- Troubleshooting
- Using other Secure Shell clients
- Appendices, including command-line tool and audit message references

For general information on Tectia Server for IBM z/OS and its features, refer to Tectia Server for IBM z/OS Product Description.

Tectia Server for IBM z/OS Administrator Manual contains instructions for installing and configuring Tectia Server for IBM z/OS on mainframes.

1.1 Documentation Conventions

The following typographical conventions are used in Tectia documentation:
Table 1.1. Documentation conventions

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

**Note**

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

**Caution**

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

### 1.1.1 Operating System Names

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

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

1.2 Customer Support

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

FAQ with how-to instructions for all Tectia products are available at 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/.

1.3 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 client tools for 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 Getting Started

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

Tectia client tools for z/OS are based on Secure Shell (SecSh) technology and they allow secure file transfer and secure system administration over an unsecured network.

Tectia client tools for z/OS can connect to any standard Secure Shell server, including Tectia Server and OpenSSH.

2.1 Product Components

Tectia client tools for z/OS consist of the following components:

- The Connection Broker: ssh-broker-g3, ssh-broker-ctl
- Secure Shell command-line tools: sshg3, scpg3, sftpg3
- Auxiliary command-line tools: ???, ssh-keygen-g3, ssh-keydist-g3, ssh-keyfetch, ssh-cmpclient-g3, ssh-secpclient-g3, ssh-certview-g3, ssh-ekview-g3
- The SOCKS Proxy: ssh-socks-proxy (see ssh-broker-g3(1)), ssh-socks-proxy-ctl (see ssh-broker-ctl(1))

2.2 Environment Variables for Client Applications

The environment variables _BPXK_AUTOCVT, _BPX_SHAREAS, and _BPX_BATCH_UMASK must be set as shown in SSHENV and sshsetenv when running Tectia client tools for z/OS programs (see below).

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:

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_BPXK_AUTOCVT=ON</td>
<td>(Required) Activates automatic text conversion of tagged UNIX file system files. If the variable is not set correctly, ssh-broker-g3 fails to start.</td>
</tr>
<tr>
<td>_BPX_SHAREAS=NO</td>
<td>(Required) Defines that ssh-broker-g3 and the client processes are run in separate address spaces.</td>
</tr>
<tr>
<td>_BPX_BATCH_UMASK=0022</td>
<td>(Required) Defines the permissions for newly created files.</td>
</tr>
<tr>
<td>SSH_DEBUG_FMT=&quot;%W(72)(2) %Dd/%Dt/%Dy %Dh:%Dm:%Ds:%Df %m/%s:%n:%f %M&quot;</td>
<td>(Optional) Can be used to specify the format of the debug messages.</td>
</tr>
<tr>
<td>_BPX_JOBLOG=STDERR</td>
<td>(Optional) Setting _BPX_JOBLOG=STDERR will cause job log messages to be written to the STDERR data set specified in the BPXBATCH job.</td>
</tr>
<tr>
<td>_EDC_ADD_ERRNO2=1</td>
<td>(Optional) Setting _EDC_ADD_ERRNO2=1 will cause errno2 to be shown in debug and error messages.</td>
</tr>
</tbody>
</table>

The Crypto Express Card (CEX) related environment variables (see Section 2.3.3) can have the following values:

- SSH_CRYPTOCARD_CIPHER_IO_THRESHOLD=<0-65536>
- SSH_CRYPTOCARD_MAC_GENERATE=YES|NO

sshsetenv:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>export _BPXK_AUTOCVT=ON</td>
<td></td>
</tr>
<tr>
<td>export _BPX_BATCH_UMASK=0022</td>
<td></td>
</tr>
<tr>
<td>export _BPX_SHAREAS=NO</td>
<td></td>
</tr>
<tr>
<td>export SSH_DEBUG_FMT=&quot;%W(72)(2) %Dd/%Dt/%Dy %Dh:%Dm:%Ds:%Df %m/%s:%n:%f %M&quot;</td>
<td></td>
</tr>
<tr>
<td>export _EDC_ADD_ERRNO2=1</td>
<td></td>
</tr>
</tbody>
</table>

(For a description of the variables, see SSHENV above.)

2.3 Running Client Programs

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 Tectia client programs is started, and stops when the client program is stopped.

It is also possible to start the Connection Broker separately from the USS command line. This way all Tectia client programs run by the user will use the same instance of **ssh-broker-g3**. See Section 2.4.1 for instructions.

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

### 2.3.1 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.

For information on the command syntax and options, see sshg3(1), scpg3(1), and sftpg3(1).

For information and examples on TCP tunneling, see Chapter 7.

For examples on remote command execution, see Section 5.2.

### 2.3.2 Under MVS

Tectia Server for IBM z/OS client-side applications can be executed in JCL by BPXBATCH, BPXBATSIL, or oshell. **scpg3** uses the same syntax for interactive and unattended file transfers. **sftpg3** has a batch mode for non-interactive file transfers.

For ease of use, file transfer client applications can also be run using a file transfer JCL procedure provided in `<HLQ>.V669.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, such as 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 Section 4.2 and Section 4.9.1.

For unattended and JCL PROC file transfer examples, see Section 6.6.2.

For using Secure Shell to run remote commands or jobs, see Section 5.2.
2.3.3 Enabling Use of IBM Crypto Express Card (CEX)

For client and socks proxy: Ciphers AES-CBC, AES-CTR and 3DES-CBC, Macs hmac-sha* are offloaded to CEX card if proper environment variables are set. CPACF will be used by default. See Tectia Server for IBM z/OS Administrator Manual Appendix H for instructions how to enable cryptographic hardware support with RACF commands.

CEX related environment variables are:

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH_CRYPTOCARD_CIPHER_IO_THRESHOLD</td>
<td>Specifies the minimum size of cipher request that will be routed to IBM cryptographic co-processor card (CEX), if the card is available. If the request size is less than the SSH_CRYPTOCARD_CIPHER_IO_THRESHOLD 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</td>
</tr>
</tbody>
</table>

If the variable is not defined, all cipher requests will use CPACF facility.

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH_CRYPTOCARD_MAC_GENERATE</td>
<td>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.</td>
</tr>
</tbody>
</table>

If the variable is not defined, all MAC requests will use CPACF facility.

2.4 Running the Connection Broker

The Connection Broker component consists of two processes:

- **ssh-broker-g3**: the Tectia Connection Broker process
- **ssh-broker-ctl**: control process for the Connection Broker. It can be used, for example, to view the status of the Connection Broker, to stop the Connection Broker, or to load private keys to memory.

Normally, there is no need to start the **ssh-broker-g3** process separately. The **sshg3**, **scpg3**, and **sftpg3** client programs start it in the on-demand mode, and stop it when the client program is stopped. However, if you reconfigure Connection Broker settings, you must stop and start the Connection Broker.

If you are running several tunneling or file transfer jobs for a single user and do not want that a separate **ssh-broker-g3** process starts each time one of these jobs is run, you can start **ssh-broker-g3** in persistent mode. After that, all client programs run by the user will use the same instance of **ssh-broker-g3**. This saves system resources. The Connection Broker can be started manually under USS or by using a JCL script. You can find an example JCL script named **SSHBRKR** in **SAMPLIB**.
2.4.1 Starting ssh-broker-g3 Manually under USS

Under USS, the Connection Broker can be started by running the following command:

```
$ /opt/tectia/bin/ssh-broker-g3
```

For information on the Connection Broker command syntax and options, see ssh-broker-g3(1).

2.4.2 Stopping ssh-broker-g3

You can stop the Connection Broker by executing one of the following commands under USS:

```
$ /opt/tectia/bin/ssh-broker-g3 --exit
```

OR:

```
$ /opt/tectia/bin/ssh-broker-ctl stop
```

If you run `ssh-broker-ctl` using a different user ID than `ssh-broker-g3`, give also the -a (alias for --broker-address) option and run `ssh-broker-ctl` as a privileged user (root).

For example, when user `SSHBRKR` owns the `ssh-broker-g3` process:

```
# /opt/tectia/bin/ssh-broker-ctl -a /tmp/ssh-SSHBRKR/ssh-broker stop
```

2.4.3 Reconfiguring ssh-broker-g3

If you make changes to the `ssh-broker-config.xml` configuration file the changes will not take effect until the Connection Broker is reconfigured by stopping and starting the Connection Broker. After you have reconfigured the Connection Broker, existing connections will continue to use the old configuration settings while new connections will use the reconfigured settings.

2.5 Connecting to a Remote Host

This section gives basic instructions on how you can log in from Tectia client tools for z/OS to a Secure Shell server with the default settings. The default settings on Tectia client tools for z/OS and Tectia Server for IBM z/OS allow login with passwords (Section 2.5.2) and public keys (Section 2.5.3).

2.5.1 Authenticating Remote Server Hosts

Remote Secure Shell servers are authenticated using either traditional public-key authentication or certificate authentication.

In public-key authentication the client trusts the server if it has a private key that matches one of the public keys in the global hostkeys directory (`/opt/tectia/etc/hostkeys`) or in the user's hostkeys directory
When a server presents a key that is not in the hostkeys directories, the user verifies the fingerprint of the remote server's public key. When the user has approved the public key, it is stored in the user's 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 (Section 2.3.2), 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.

Optionally, the ssh-broker-ctl probe-key or ssh-keyfetch tool can be used for storing the remote server keys before the connections are made. See ??? and ssh-keyfetch(1) respectively.

For more information on server authentication using public keys, see Section 4.2.

2.5.2 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 client tools for 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 (see Section 4.4.1), or preferably use public-key authentication and a private key without a passphrase (see Section 4.5).

For more information on password authentication, see Section 4.4.
2.5.3 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 (ssh-keygen-g3(1)), and they can be distributed to the remote hosts using the ssh-keydist-g3 tool (ssh-keydist-g3(1)).

For more information and examples on public-key authentication, see Section 4.5.

2.5.4 Logging in with Command-Line sshg3

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

1. Enter the sshg3 command using the following syntax:

   
   $ sshg3 <hostname>

   For example:

   $ sshg3 abc.example.com

   The basic syntax is:

   $ sshg3 user@host#port

   where:

   • user - Enter a user name that is valid on the remote host. The user attribute is optional. If no user name is given, the local user name is assumed.

   • host - Enter the name of the remote host as an IP address, FQDN (fully qualified domain name), or short host name. The remote host must be running a Secure Shell version 2 server.

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

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

   
   $ sshg3 profile1

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

   For more information on the sshg3 commands and options, see sshg3(1).
2. The server authentication phase starts. The server sends its public key to the client for validation (when server public-key authentication is used).

Tectia client tools for z/OS checks if this key is already stored in your own host key directory. If not, the host key directory common to all users on your computer is checked next.

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

When Tectia client tools for z/OS receives a new host public key, a host identification message is displayed. For example:

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

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

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

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

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

For more information on server authentication, see Section 4.2.

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

After the server has successfully authenticated you, the Secure Shell connection to the server is opened.
Chapter 3 Configuring Client Tools

This chapter gives instructions on the basic configuration settings of the Tectia client components on z/OS.

The authentication and connection profile settings for Tectia client tools for z/OS are made in the Connection Broker configuration, because the Connection Broker handles all cryptographic operations and authentication-related tasks for Tectia client tools for z/OS.

3.1 Client Configuration Files

The following files located in the /opt/tectia/etc directory are used to store the client configuration information:

- /opt/tectia/etc/ssh-broker-config.xml: the global Connection Broker configuration file
- /opt/tectia/etc/ssh_ftadv_config: the global file transfer advisor profile configuration file
- /opt/tectia/etc/ssh-socks-proxy-config.xml: the global SOCKS Proxy configuration file
- /opt/tectia/etc/hostkeys: the global directory for known remote server host keys

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

- $HOME/.ssh2/ssh-broker-config.xml: the user-specific Connection Broker configuration file
- $HOME/.ssh2/ssh_ftadv_config: the user-specific file transfer advisor profile configuration file
- $HOME/.ssh2/ssh-socks-proxy-config.xml: the user-specific SOCKS Proxy configuration file
- $HOME/.ssh2/hostkeys: the user-specific directory for known remote server host keys
- $HOME/.ssh2/identification: the identification file used with public-key authentication
- $HOME/.ssh2/random_seed: the random number seed file for cryptographic operations
3.1.1 Editing the Configuration Files

The default location for the configuration files is `/opt/tectia/etc` for the global configurations and `$HOME/.ssh2` for the user-specific configuration files. You can edit the `ssh-broker-config.xml` and `ssh-socks-proxy-config.xml` files with your favorite XML or text editor.

The Connection Broker configuration file `ssh-broker-config.xml` is a valid XML file. For a detailed description of the Connection Broker configuration options, see `ssh-broker-config(5)`. For a quick reference to the configuration options, see Section A.2.

The SOCKS Proxy configuration file `ssh-socks-proxy-config.xml` is a valid XML file. For more information on the SOCKS Proxy configuration options for transparent tunneling, see `ssh-broker-config(5)` and `Tectia Server for IBM z/OS Administrator Manual`.

The `ssh_ftadv_config` configuration file contains file transfer profiles used by `scpg3` and `sftpg3`. The file does not exist by default, but can be enabled by copying the example profile file `/opt/tectia/etc/ssh_ftadv_config.example` to `/opt/tectia/etc/ssh_ftadv_config` (globally for all users) or to `$HOME/.ssh2/ssh_ftadv_config` (for a specific user). For more information on the file transfer advisor configuration options, see Section 9.4.2.

3.2 Environment Variables

In addition to the configuration file, also environment variables can be used to configure the clients. Environment variables override values specified in the configuration file.

For a list of file transfer environment variables, see Section 6.4.2.

3.3 Command-Line Options

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

For a list of the available options, see `sshg3(1), scpg3(1), and sftpg3(1)`.
Chapter 4 Authentication

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

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

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

User authentication methods used by the Tectia client on z/OS by default are: public-key, keyboard-interactive, and password authentication. In addition, the client supports host-based authentication.

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

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

<table>
<thead>
<tr>
<th>Authentication method</th>
<th>Tectia Server</th>
<th>Tectia Client, ConnectSecure, and client tools on Server for IBM z/OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unix(^a)</td>
<td>Windows (x)</td>
</tr>
<tr>
<td>Password</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>Public-key</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>Certificate</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>Host-based</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>Keyboard-interactive</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>PAM(^c)</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>RSA SecurID(^c)</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>RADIUS(^c)</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>GSSAPI/Kerberos</td>
<td>(x)</td>
<td>(x)</td>
</tr>
</tbody>
</table>

\(^a\) Including Tectia Server for Linux on IBM System z.
\(^b\) Including certificates in files and SAF certificates.
\(^c\) Through keyboard-interactive.

4.2 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.

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

```
Host key for the host "alpha" not found from database.

The fingerprint of the host public key is:
"xicec-kahog-kyih-fufib-nyzed-logum-pycuz-besug-galoh-gupah-pyxax*

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

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

Never save a host public key without verifying its authenticity!

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

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

After verifying the fingerprint, it is safe to continue connecting. Relevant information about the server public key will then be stored on the client-side machine. With Tectia client on z/OS, it is stored in the user's $HOME/.ssh2/hostkeys USS directory.

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

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

4.2.1 Host Key Storage Formats

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

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

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

```xml
<known-hosts path="$HOME/.ssh2/hostkeys" filename-format="plain" />
```
If you are adding the keys manually, the keys should be named with the `key_<port>_<host>.pub` pattern, where `<port>` is the port the Secure Shell server is running on and `<host>` is the host name you use when connecting to the server (for example, `key_22_alpha.example.com.pub`).

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

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

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

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

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

- `sshg3 user@alpha`

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

- `sshg3 user@alpha.example.com`

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

- `sshg3 user@10.1.101.10`

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

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

  produces the key with IPv6 address (in plain format `key_22_fd0000100001010300000000000012f69.pub`)

When connecting to a server using its IPv6 address, the IPv6 address given to Tectia client tools for z/OS is canonicalized without the colons, and the canonical format is used in the known host key file name. For example, the plain format host key file for `::1#10022` would be `key_10022_0000000000000000000000000000000001.pub`. The canonical format is also used in the process of saving and reading hashed host keys.

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

© 2007–2020 SSH Communications Security Corporation

Tectia® Server 6.6 for IBM z/OS User Manual
After the first connection, the locally stored information about the server public key will be used in server authentication.

4.2.2 Using the System-Wide Host Key Storage

If a host key is not found in the user-specific host key directory, it is next searched from the /opt/tectia/etc/hostkeys directory. Host key files are not automatically put in the system-wide directory but they have to be updated manually by the system administrator.

Storing Keys in the Hashed Format

To obtain and store hashed remote host keys in the system-wide storage you can either copy the keys manually from the server to the client or you can use the ssh-keydist-g3 tool from the client machine.

To copy the keys manually:

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

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

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

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

3. Connect with Tectia client tools for z/OS to the remote server, verify the fingerprint, and save the key.

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

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

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

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

After creating the system-wide location for host keys, you can maintain it by using the ssh-keygen-g3 tool.
The following copy examples show the most frequently needed commands for host key storage maintenance. The commands use the user-specific hostkey storages ($HOME/.ssh2/hostkeys and possibly the $HOME/.ssh/known_hosts file) as the source. If keys are to be copied from a different source, you need to append an appropriate --hostkeys-directory or --hostkey-file option to the command.

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

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

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

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

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

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

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

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

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

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

To store the keys using ssh-keydist-g3:

1. Run ssh-keydist-g3 with the -g (alias for --accept-hostkeys-globally) option as a privileged user on the client, for example:

   ```
   # ssh-keydist-g3 -N -i -g -A /tmp/newkeys.log host1 host2 host3
   ```

   The -N option specifies that new host keys are accepted, and -A specifies a log file (in this example /tmp/newkeys.log) listing the accepted new host keys.

   Substitute the appropriate list of host names as the command arguments.

   The -i option specifies that the host keys are also stored using the IP addresses of the hosts. Transparent FTP tunneling and FTP-SFTP conversion require that the keys are stored using the IP address. However, the host keys are not automatically stored using the long host name. If you want to do also that, specify the long host name in addition to the short host name as an argument for ssh-keydist-g3.

2. After the transfer, verify the fingerprints of the keys from the log file /tmp/newkeys.log.

For more information on the ssh-keydist-g3 options, see ssh-keydist-g3(1).
Storing Keys in the Plain Format

To obtain and store traditional (plain) remote host keys in the system-wide storage you can either copy the keys manually from the server to the client or you can use the `ssh-keydist-g3` tool from the client machine.

To copy the keys manually:

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

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

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

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

   On the client:

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

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

To store the keys using `ssh-keydist-g3`:

1. Run `ssh-keydist-g3` with the `-F plain` (-F is equal to `--accepted-host-key-filename-format`) and `-g` options as a privileged user on the client, for example:

   ```
   # ssh-keydist-g3 -N -F plain -g -A /tmp/newkeys.log host1 host2 host3
   ```

   Substitute the appropriate list of host names as the command arguments. In the example above, the following host keys are fetched:

   ```
   key_22_host1.pub
   key_22_host2.pub
   key_222_host3.pub
   ```

2. After the transfer, verify the fingerprints of the keys from the log file `/tmp/newkeys.log`.

For more information on the `ssh-keydist-g3` options, see `ssh-keydist-g3(1)`.

4.2.3 Resolving Hashed Host Keys

Tectia client tools for z/OS includes a tool to resolve which hashed host key belongs to which server. As there can be several server host keys stored on the client-side host, and the file name does not show the server name, it is sometimes necessary to check if a certain server public key is stored on the client host.
On the command line, the command syntax is:

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

For example:

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

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

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

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

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

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

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

### 4.2.4 Using the OpenSSH known_hosts File

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

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

```xml
<general>
  ...
  <known-hosts path="/home/username/.ssh/known_hosts" />
  <known-hosts path="/etc/ssh/ssh_known_hosts" />
  <known-hosts path="/home/.ssh2/hostkeys" />
  <known-hosts path="/u/username/.ssh2/hostkeys" />
</general>
```
You can disable OpenSSH `known_hosts` file handling by defining an empty setting: `known-hosts path=""`. After this, only the Tectia-related hostkey directories will be used.

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

### 4.3 Server Authentication with Certificates

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

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

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

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

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

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

   **Caution**

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

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

4. The client verifies that the server has a valid private key by checking the signature in the initial packet.
During authentication the system checks that the certificate has not been revoked. This can be done either by using the Online Certificate Status Protocol (OCSP) or a certificate revocation list (CRL), which can be published either in an LDAP or HTTP repository.

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

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 Tectia validation can use CA certificates stored in file (Section 4.3.1) or in SAF (Section 4.3.2), or the SAF validation can be used alone (Section 4.3.3).

**SAF Validation**

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 under `general/known-hosts/key-store` if (only) SAF certificate checking is to be used. If at least one of these elements is configured, SAF validation will be used and Tectia validation will not 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 under `general/cert-validation/key-store`, the Tectia Certificate Validator takes its trusted CA certificates from SAF, otherwise they are read from files.

### 4.3.1 CA Certificates Stored in File

To configure the client to trust the server's certificate by using CA certificates stored in a file, perform the following tasks. Replace the names and IDs with those appropriate to your system:

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

   Certificates can be extracted from a PKCS #7 package by specifying the `-7` option with `ssh-keygen-g3`. 
2. Define the CA certificate(s) to be used in host authentication in the ssh-broker-config.xml file under the general element:

```xml
<cert-validation end-point-identity-check="yes"
    socks-server-url="socks://fw.example.com:1080">
  <ldap-server address="ldap://ldap.example.com:389" />
  <ocsp-responder url="http://ocsp.example.com:8090" validity-period="0" />
  <ca-certificate name="ssh_ca1"
      file="ssh_ca1.crt"
      disable-crls="no"
      use-expired-crls="100" />
</cert-validation>
```

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

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

**Note**

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

Define also the LDAP server(s) or OCSP responder(s) used for CRL checks. If the CA services (OCSP, CRLs) are located behind a firewall, define also the SOCKS server.

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

3. Setting the certificate authentication method either under default settings (default-settings/server-authentication-methods) or per connection profile (profiles/profile/server-authentication-methods) defines that the server must authenticate with a certificate or else the authentication will fail.

```xml
<server-authentication-methods>
  <auth-server-certificate />
</server-authentication-methods>
```

For more information on the configuration file options, see ssh-broker-config(5).

### 4.3.2 CA Certificates Stored in SAF

To configure the client to trust the server's SAF certificate by using Tectia validation, perform the following tasks. Replace the names and IDs with those appropriate to your system:

1. Get the CA certificate and store it to a data set, for example 'HOSTCA.CRT'.

2. To add the CA certificate into SAF, give the following TSO commands:
3. For the settings to take effect, give the following TSO command:

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

4. Define the z/OS SAF external key provider that contains the CA certificates in the `general/cert-validation/key-store` element:

```bash
<cert-validation end-point-identity-check="yes"
  socks-server-url="socks://fw.example.com:1080"
  ldap-server address="ldap://ldap.example.com:389" />
<ocsp-responder url="http://ocsp.example.com:8090" validity-period="0" />
<key-store type="zos-saf"
  init="KEYS(ID(SSHD2) RING(SSH-HOSTCA)) TRUST-ANCHORS"
  disable-crls="no"
  use-expired-crls="0" />
</cert-validation>
```

Define also the LDAP server(s) or OCSP responder(s) used for CRL checks. If the CA services (OCSP, CRLs) are located behind a firewall, define also the SOCKS server.

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

5. Setting the certificate authentication method either under default settings (default-settings/server-authentication-methods) or per connection profile (profiles/profile/server-authentication-methods) defines that the server must authenticate with a certificate or else the authentication will fail.

```bash
<server-authentication-methods>
  <auth-server-certificate />
</server-authentication-methods>
```

For more information on the configuration file options, see `ssh-broker-config(5)`. For information on the format of the external key initialization string, see the section called “Key Store Configuration Examples”.

### 4.3.3 Server Certificates Stored in SAF

**Note**

If there is at least one `general/known-hosts/key-store` element configured, SAF validation will be used and Tectia validation will not be used. This is different from Tectia Server for IBM z/OS.
version 5.x, where you could specify *saf,tectia*, and both validations were performed and both had to succeed.

To configure the client to trust the server's SAF certificate by using SAF validation only, perform the following tasks. Replace the names and IDs with those appropriate to your system:

1. Get the server host certificate and store it to a data set, for example `SERVER1.CRT`.

2. To add the server certificate into SAF, give the following TSO commands:

   ```
   RACDCERT ID(USER) ADD('SERVER1.CRT') TRUST WITHLABEL('SERVER1')
   RACDCERT ID(USER) ADDRING(SSH-HOSTKEYS)
   RACDCERT ID(USER) CONNECT(ID(USER) LABEL('SERVER1') RING(SSH-HOSTKEYS) USAGE(PERSONAL))
   RACDCERT ID(USER) LISTRING(SSH-HOSTKEYS)
   ```

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

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

4. Define the z/OS SAF external key provider that contains the server host certificates in the `general/known-hosts/key-store` element:

   ```
   <known-hosts>
   ...
   <key-store type="zos-saf"
     init="KEYS(ID(USER) RING(SSH-HOSTKEYS))" />
   </known-hosts>
   ```

For more information on the configuration file options, see `ssh-broker-config(5)`. For information on the format of the external key initialization string, see the section called “Key Store Configuration Examples”.

### 4.4 User Authentication with Passwords

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

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

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

Other authentication methods can be listed in the configuration file as well. Place the least interactive method first (password is usually the last one).
4.4.1 Password Stored in a File or Data Set

Password authentication normally requires user interaction. For situations where user interaction is not possible, for example when running Tectia client programs from JCL, you can have the password stored in a file or data set.

With sshg3, scp3, and sftp3, use the `--password-file://FILE` option to provide the password. With ssh-keydist-g3, use the `--password-file FILE` option to provide the password.

**Note**

When storing a password in a file or data set, make sure that the access permissions are correct.

For non-interactive batch jobs, we recommend that you use public-key authentication without a passphrase, or host-based authentication. These methods provide more security than a password stored in a file.

**Password Stored in a File**

To set up authentication with a password stored in a file:

1. Create a file, for example `/home/userid/passwd_file`.
2. The file must be readable to the user that created it only:
   
   ```sh
   $ 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:
   
   ```
   MyPassS
   ```

   To use the password stored in a file, for example with sftp3, run the following:

   ```sh
   $ sftp3 --password-file:///home/userid/passwd_file
   ```

**Password Stored in a Data Set**

To set up authentication with password stored in a data set:

1. Allocate a data set or a data set member, for example:
   
   ```
   //USERID.PASSWD
   ```
2. Make sure that the data set is accessible only by the correct user ID.
3. Edit the password data set to contain your password on the remote system. The format of the password data set is one line containing only the password. For example:
To use the password stored in a data set, for example with `sftpg3`, run the following:

```bash
$ sftpg3 --password=file://"///USERID.PASSWD"
```

### 4.5 User Authentication with Public Keys in a File

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

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

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

⚠️ **Caution**

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

To use public-key authentication with Tectia client tools for z/OS, do the following actions:

1. Generate a key pair with `ssh-keygen-g3` (see Section 4.5.1).
2. Upload your public key to the remote host computer (see Section 4.5.2).

For instructions on using public-key authentication to connect from other hosts to Tectia Server for IBM z/OS, see Section 9.1.

In the instructions in the following sections,

- **Server** is the remote host running the Secure Shell server that you are trying to connect to.
- **ServerUser** is the user name on **Server** that you are logging in as.
- **Client** is the host running the Secure Shell client (Tectia client tools for z/OS).
- **ClientUser** is the user name on **Client** that should be allowed to log in to **Server** as **ServerUser**.
The instructions assume that ClientUser is allowed to log in to Server as ServerUser using some other authentication method (usually password).

### 4.5.1 Creating Keys with ssh-keygen-g3 on z/OS

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

```bash
$ ssh-keygen-g3
Generating 2048-bit rsa key pair
   9 oOo.oOo.oOo
Key generated.
2048-bit rsa, ClientUser@Client, Wed Feb 2 2016 12:09:46 +0200
Passphrase :
Again :
Private key saved to /home/ClientUser/.ssh2/id_rsa_2048_a
Public key saved to   /home/ClientUser/.ssh2/id_rsa_2048_a.pub
```

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

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

The key pair is by default stored in your `$HOME/.ssh2` directory (created by **ssh-keygen-g3** if it does not exist previously).

In the example above, the private key file is `id_rsa_2048_a`. The public key file is `id_rsa_2048_a.pub`, and it can be distributed to other computers.

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

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

For more information on the **ssh-keygen-g3** options, see [ssh-keygen-g3(1)](http://www.example.com/ssh-keygen-g3).
4.5.2 Uploading Public Keys from z/OS to Remote Host

All commands in this section are shown using $sshg3$ and $scpg3$ from the machine running Tectia client tools for z/OS. Server-side configuration can also be done by logging in to the remote server and entering the commands locally.

To enable public-key authentication with your key pair:

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

   By default, the Connection Broker attempts to use each key found in the $HOME/.ssh2$ directory.

   You can also add other directory locations for keys using the $general/key-stores/key-store$ element in the $ssh-broker-config.xml$ file. See the section called “Key Store Configuration Examples”.

2. (Optional) Create an identification file.

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

   Create a file called $identification$ in your $HOME/.ssh2$ directory, for example:

   ```
   $ cd $HOME/.ssh2
   $ echo "IdKey id_rsa_2048_a" >> identification
   ```

   You now have an identification file that consists of one line that denotes the file containing your private key:

   ```
   IdKey id_rsa_2048_a
   ```

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

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

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

   - Tectia Server on Unix or z/OS:

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

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

     ```
     $ sshg3 ServerUser@tectia_unix mkdir .ssh2/authorized_keys
     ```
• Tectia Server on Windows:

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

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

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

• OpenSSH server on Unix or z/OS:

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

4. Copy the public key to Server.

Keys created with `ssh-keygen-g3` on z/OS are stored in the EBCDIC format. When the public key is transferred to a Unix or Windows server, it must be converted to ASCII format. This can be done by specifying the `-a` option with `scp43`.

If public-key authentication is configured between mainframe servers, conversion is not needed.

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

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

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

```bash
$ scp43 -a id_rsa_2048_a.pub ServerUser@tectia_server:.ssh2/authorized_keys/
```

• Tectia Server 4.x on Unix and Windows:

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

```bash
$ scp43 -a id_rsa_2048_a.pub ServerUser@tectia4x_server:.ssh2/
```

• Tectia Server for IBM z/OS:

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

```bash
$ scp43 id_rsa_2048_a.pub ServerUser@tectia_zos:˜/.ssh2/
```

• OpenSSH server on Unix:

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

```bash
$ scp43 -a id_rsa_2048_a.pub ServerUser@open_unix:.ssh/
```
• OpenSSH server on z/OS:

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

```bash
$ scp g3 -a id_rsa_2048_a.pub ServerUser@open_zos:.ssh/
```

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

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

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

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

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

```bash
$ sshg3 ServerUser@tectia_server "echo Key id_rsa_2048_a.pub >> .ssh2/authorization"
```

On a Windows server:

```bash
$ sshg3 ServerUser@tectia4x_win "cmd /c echo Key id_rsa_2048_a.pub >> .ssh2/authorization"
```

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

```
Key id_rsa_2048_a.pub
```

This directs Tectia Server to use id_rsa_2048_a.pub as a valid public key when authorizing your login.

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

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

```bash
$ sshg3 ServerUser@open_server "ssh-keygen -i -f .ssh/id_rsa_2048_a.pub >> .ssh/authorized_keys"
```

6. Make sure that public-key authentication is enabled in the ssh-broker-config.xml file (it is enabled by default).
Other authentication methods can be listed in the configuration file as well. Place the least interactive method first.

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

Try to log in:

```
Client$ sshg3 Server
```

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

### 4.5.3 Using Keys Generated with OpenSSH

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

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

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

The public key can be uploaded to the server the same way as with standard SSH2 keys. See Section 4.5.2.

### 4.6 User Authentication with Certificates

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

In brief, certificate authentication works as follows:
1. The client sends the user certificate (which includes the user's public key) to the server. The packet also contains data unique to the session and it is signed by the user's private key.

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

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

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

The Tectia Server for IBM z/OS client programs use SAF certificates when the configuration includes certificate authentication and a private key provider. The configuration specifies which keys and certificates the client will offer.

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` option 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.

### 4.6.1 Certificates Stored in File

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

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

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

   ```
   $ ssh-cmpclient-g3 INITIALIZE
   -P generate://ssh2:passphrase@rsa:2048/user_rsa \\
   -o /home/user/.ssh2/user_rsa -p 62154:ssh \\
   -s 'C=FI,O=SSH,CN=user;email=user@example.org' \\
   'C=FI, O=SSH, CN=Test CA 1'
   ```

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

   By default, the Connection Broker attempts to use each key found in the `$HOME/.ssh2` directory.

   You can also add other directory locations for keys using the `general/key-stores/key-store` element in the `ssh-broker-config.xml` file. See the section called “Key Store Configuration Examples”.

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

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

```
CertKey user_rsa
```

The certificate itself will be read from user_rsa.crt.

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

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

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

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

4.6.2 Certificates Stored in SAF

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

1. To create a user key in SAF, give the following TSO commands:

   ```
   RACDCERT ID(USER) GENCERT SUBJECTSDN(CN('User') OU('RD') O('EXAMPLE'))
   SIZE(1024) WITHLABEL('USER')
   RACDCERT ID(USER) LIST
   ```

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

   ```
   RACDCERT ID(USER) GENREQ(LABEL('USER')) DSN('USER.CRT.REQ')
   ```

3. Use the PKCS #10 certification request in the data set 'USER.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 'USER.CRT'.

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

   ```
   RACDCERT ID(USER) ADD('USER.CRT') TRUST WITHLABEL('USER')
   RACDCERT ID(USER) ADDRING(USER)
   ```
6. For the settings to take effect, give the following TSO command:

```
SETROPTS RACLST(DIGTCERT) REFRESH
```

7. Define the z/OS SAF external key provider and its initialization string with the `general/key-stores/key-store` element in the `ssh-broker-config.xml` file:

```
<key-stores>
  <key-store type="zos-saf"
    init="KEYS(ID(%U) RING(%U))" />
</key-stores>
```

The initialization string 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

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

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

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

For more information on the configuration file options, see `ssh-broker-config(5)`. For information on the format of the external key initialization string, see the section called “Key Store Configuration Examples”.

### 4.7 Host-Based User Authentication

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

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

```
4.8 User Authentication with Keyboard-Interactive

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

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

Note

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

To enable keyboard-interactive authentication on Tectia client tools for z/OS, make sure that you have the following line in the ssh-broker-config.xml file:

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

4.9 Distributing Public Keys Using the 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 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.

You can use `/opt/tectia/bin/ssh-broker-ctl` to fetch remote-host keys. Then use the key-distribution tool `/opt/tectia/bin/ssh-keydist-g3` for storing multiple remote host keys to user-specific or common key store and setting up public-key authentication to multiple hosts.

For more information about the `ssh-broker-ctl probe-key` and `ssh-keydist-g3` options, see ?? and `ssh-keydist-g3(1)` respectively.

Most of the examples in this section are executed from Unix shell (for example, OMVS shell), but the same commands can also be run in JCL using `BPXBATCH`. 
4.9.1 Fetching Remote Server Keys

Tectia client tools for z/OS 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/tectia/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 4.2.1.

Note

We recommend using ssh-broker-ctl for fetching remote server keys. ssh-keydist-g3 key fetching is being obsoleted.

Examples of Fetching Remote Server Keys

The following examples illustrate using commands ssh-keydist-g3 and ssh-broker-ctl probe-key for fetching remote server host keys.

Caution

When ssh-keydist-g3 is run with the -N (alias for --accept-host-keys) 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. Similar risk is also applied to ssh-broker-ctl probe-key.

Example 1: ssh-keydist-g3 Using USS

This example is run under the 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: ssh-keydist-g3 Using JCL

This example HOSTSAVE from <HLQ>.V669.SAMPLIB presents a JCL script that does the same steps as the USS command in Example 1 above (the options are given in short format):

```
//HOSTSAV EXEC PGM=BPXBATS,REGION=0M,TIME=NOLIMIT
//STDPARM DD *
```
Example 3: ssh-broker-ctl Using USS

For example, adding the host key of server.example.com:

```
$ ssh-broker-ctl probe-key --add-hostkey server.example.com#22
```

You may provide the expected fingerprint of the host key with --hostkey-fp. In this case, the host key is only added if its fingerprint matches what you provided with --hostkey-fp:

```
$ ssh-broker-ctl probe-key \
--hostkey-fp=5ffee87e2469e4e2025ff6893db1e31f9b6f32ae \
--add-hostkey server.example.com#22
```

Server hostkey: 256 bit ed25519 key
SHA-1: 5f7f2e87e2469e4f6025ff6893db1e31f9b6f32ae
Server hostkey saved.

Example 4: ssh-broker-ctl Using JCL

You can also use ssh-broker-ctl via JCL, similar to the following:

```
//HOSTSAVF JOB ,,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1), 
   NOTIFY=&SYSUID 
//* 
//* HOSTSAF - Store host keys of remote Secure Shell servers 
//* 
//* Copyright (c) 2019 SSH Communications Security Corporation. 
//* This software is protected by international copyright laws. 
//* All rights reserved. 
//* 
//* This example uses the ssh-broker-ctl utility to fetch and 
//* store the host keys of remote Secure Shell servers. 
//* 
//* This example: 
//* - Connects to remote Secure Shell servers host1, host2, and host3 
//* and saves the host keys in plain format under the user's USS 
//* home directory $HOME/.ssh2/hostkeys/. The host keys are also 
//* saved using the IP addresses of the hosts. 
//* 
//* Required SSHENV environment variables are defined using 
//* the STDENV DD card. 
//* Modify the DD statement according to your environment. 
```
Note

If the fetching of host keys fails, make sure you have the environment variable `SSH_CHARSET_CONV` correctly set to `SSH_CHARSET_CONV="/opt/tectia/lib/shlib/i18n_iconv.so"`.

4.9.2 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 Tectia clients from a Telnet or Secure Shell session. They can also use public-key pairs with a null passphrase if they want to run the 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 `ssh-keygen-g3` is run with the `-P` option, which requests a null passphrase, it can be run in JCL. It must be run under the batch user's user ID in order for the file permissions to be set properly. For more information on the `ssh-keygen-g3` options, see `ssh-keygen-g3(1)`.

The batch user accesses the remote machine using an account created and administered on the remote machine. The remote user name may either be the same as the batch user's RACF user ID, or the same but in lower case, or a different user name. 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 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 data set or in a file. See the section called “Password Stored in a File” and the section called “Password Stored in a Data Set” for instructions. The file name 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 user name, the remote host DNS name or IP address, and the type of the remote Secure Shell server (Tectia Server on Unix, Tectia Server on Windows, Tectia Server for IBM z/OS on mainframe, or OpenSSH on Unix).

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 <HLQ>.V669.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=BPXBATSU,REGION=0M,TIME=NOLIMIT
//STDPARM DD *
PGM /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
//STDEVN DD DSN=<HLQ>.V669.PARMLIB(SSHENV),DISP=SHR
//STDOUT DD SYSOUT=* 
//STDERR DD SYSOUT=* 
//STDIN  DD DUMMY 
```
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 (Tectia Unix, Tectia Windows, Tectia z/OS, or 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 System Administration

Secure system administration is a common use case for Secure Shell.

The server settings may limit the services that are accessible via Secure Shell. For example, the Secure Shell server may deny terminal access and allow only file transfer or tunneling. The restrictions from the server perspective are described in the Tectia Server for IBM z/OS Administrator Manual.

5.1 Defining Shell Access

5.1.1 Setting Codepage for Remote Account

By default Tectia Server for IBM z/OS uses IBM-1047 as the codepage when connecting a user to an application. The application is typically /bin/sh. For interoperability with diverse SSH client programs, Tectia uses ISO8859-1 on the line.

You can change the codepage with command chcp. If the session on the server is started up with a terminal and with ShellConvert=yes, the chcp command can be used to change the Transfer CCS and the Account CCS during that session. However, it is not possible to change to using a translation table or to change the line delimiters.

The codepage change happens on the outbound stream starting from the output of the chcp command and on the next prompt. On the inbound stream, the codepage is changed starting with the character following the character that ends the command.
5.2 Running Remote Commands

sshg3 can be used to run remote commands or jobs.

For example, to display the operating system version and other details of a remote host, execute the following command:

```
$ sshg3 user1@host1.example.com "uname -a"
```

More example commands are shown in the following sections.

5.2.1 Remote Command Examples from USS

**Example 1:** Remote command for creating directory `testdir` on a remote Unix system:

```
$ sshg3 user1@unix.example.com mkdir testdir
```

**Example 2:** Remote command for creating a directory on a remote Windows system:

```
$ sshg3 user2@windows.example.com "cmd /c mkdir testdir"
```

Note that quotes ("*) are needed around remote Windows commands when they are run from the USS shell.

**Example 3:** Multiple commands can be separated using a semicolon:

```
$ sshg3 user1@unix.example.com "cd /tmp; ls -l test.*; rm test.txt"
```

Note that quotes ("*) are needed around multiple remote commands when they are run from the USS shell.

5.2.2 Remote Command Examples using JCL

**Example 1:** JCL with remote command for creating a directory on a remote Unix system:

```
//SSH EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT
//STDPARM DD *
PGM /opt/tectia/bin/sshg3 user1@unix.example.com
mkdir testdir
//STENV DD DSN=<HLQ>.V669.PARMLIB(SSHENV),DISP=SHR
//STDOUT DD SYSOUT=* 
//STDERR DD SYSOUT=* 
//STDIN DD DUMMY
//
```

(Use this JCL to complete the following examples.)

**Example 2:** Remote command for creating a directory on a remote Windows system. Use the following as input data for the STDPARM DD in Example 1:

```
PGM /opt/tectia/bin/sshg3 user2@windows.example.com
cmd /c mkdir testdir
```
Note that quotes (*) are not needed when commands are run from JCL.

**Example 3:** Multiple commands. Use the following as input data for the STDTPARM DD in Example 1:

```bash
PGM /opt/tectia/bin/sshg3 user1@unix.example.com
cd /tmp;
ls -l test.*;
rm test.txt
```

Note that quotes (*) are not needed around multiple remote commands when they are run from JCL.

## 5.3 Managing JCL Jobs over SFTP

Tectia Server for IBM z/OS provides the functionality for managing JCL jobs remotely. The JCL scripts are transferred to the z/OS MVS Job Entry Subsystem (JES) using SFTP. The JES interface of Tectia Server for IBM z/OS provides the following functions:

- Submitting a job
- Receiving the spool output of a job
- Deleting jobs
- Displaying the status of all of the user’s jobs

Either a file transfer advice string (FTADV) or the SFTP `site` command is required to interface with JES instead of the file system. For more information on the FTADV/site parameters, see Section 6.4.1.

In the following sections, examples are presented on how to manage JCL jobs over SFTP using three different clients: Tectia `sftpg3`, Tectia `scpg3` and OpenSSH `sftp`.

### 5.3.1 Tectia sftpg3

#### Submitting a Job

The following examples show how to submit a job for execution and receive a notification of the ID assigned to the submitted job. In the examples it is assumed that the JCL script `br14.jcl` with the following contents is stored in the directory `/home/user1/src/jcl/` on a Unix host:

```sh
//USERJ0  JOB,,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
//                    NOTIFY=&SYSUID
//*
//STEP00  EXEC PGM=IEFBR14
//
```

**Example 1.** Interface with JES using a file transfer advice string:
To retrieve the spool output of a submitted job, run the get command with the job’s ID.

**Example 1.** Interface with JES using a file transfer advice string to specify the file type and code set conversion:

```
sftp> get /ftadv:filetype=jes,C=ISO8859-1,D=IBM-1047/JOB03198
JOB03198                           |  1.1kB |     431B/s | TOC: 00:00:02 | 100%
```

**Example 2.** Interface with JES using the site command:

```
sftp> site filetype=SEQ
```

Retrieving the Spool Output of a Job

To retrieve the spool output of a submitted job, run the get command with the job’s ID.

**Example 1.** Interface with JES using a file transfer advice string to specify the file type and code set conversion:

```
sftp> get /ftadv:filetype=jes,C=ISO8859-1,D=IBM-1047/JOB03198
JOB03198                           |  1.1kB |     431B/s | TOC: 00:00:02 | 100%
```

**Example 2.** Interface with JES using the site command:

```
sftp> site filetype=SEQ
```

© 2007–2020 SSH Communications Security Corporation
Set file transfer mode to ASCII.

Run the site command to change file type to JES.

Retrieve the job.

Return to accessing the normal file system.

Deletion Jobs

To delete a job (in this example JOB03198), use the rm command and an advice string to set file type to JES:

```
sftp> rm /ftadv:filetype=JES/JOB03198
```

Listing Jobs

**Example 1.** To display the status of all the jobs that are on the JES spool for your user ID, enter the following command:

```
sftp> ls /ftadv:filetype=JES/
```

**Example 2.** To list jobs in the long name format, enter:

```
sftp> ls -l /ftadv:filetype=JES/
```

**Example 3.** To list the contents of a specific job (in this example JOB03419) in the long name format:

```
sftp> ls -l /ftadv:filetype=JES/JOB03419/
```

Note the required trailing slash after the job ID.

5.3.2 Tectia scp3

Submitting a Job

**Example 1.** Copy br14.jcl to mf_server using the -a option to enable automatic ASCII-EBCDIC conversion:

```
scp3 -a /home/user1/src/jcl/br14.jcl mf_server#22:/ftadv:filetype=JES/
```

**Example 2.** If -a is not used, code set conversion must be specified in the advice string (in this example the code set during the transfer is ISO8859-1 and the server should store the data set with the IBM-1047 code set):

```
```
Retrieving the Spool Output of a Job

Retrieve the spool output for JOB03390, performing automatic code set conversion with the \texttt{-a} option:

\texttt{scp3g3 \texttt{-a} \textit{mf}_server\#22:/ftadv:filetype=JES/JOB03390}.

5.3.3 OpenSSH sftp

Submitting a Job

Submit a job using \texttt{put} and a file transfer advice string (\texttt{/ftadv:filetype=JES,C=ISO8859-1,D=IBM-1047/}) to set file type to JES and specify code set conversion. In this example the code set is ISO8859-1 during the transfer and the server should store the data set with the IBM-1047 code set:

\texttt{put /home/user1/src/jcl/br14.jcl /ftadv:filetype=JES,C=ISO8859-1,D=IBM-1047/}

Or, alternatively:

\texttt{cd /ftadv:filetype=JES/ ❶}

\texttt{put /home/user1/src/jcl/br14.jcl ./ftadv:C=ISO8859-1,D=IBM-1047/ ❷}

Deleting Jobs

Example 1. To delete a job (JOB01234), run \texttt{rm} and use an advice string to set file type to JES:

\texttt{rm /ftadv:filetype=jes/JOB01234}
**Example 2.** To delete all jobs with IDs that begin with JOB012, enter the following command (the wildcard * is escaped with a backslash (\) in order to leave file name globbing to the server):

```
rm /ftadv:filetype=jes/JOB012\*
```

When you issue the **rm** command on a job, all spool output related to the job is deleted.

**Listing Jobs**

**Example 1.** To list jobs in the long name format, enter:

```
ls -l /ftadv:filetype=JES,C=ISO8859-1,D=IBM-1047/
```

Or, alternatively:

```
cd /ftadv:filetype=JES,C=ISO8859-1,D=IBM-1047/
ls
ls -l
```

**Example 2.** To list the spool output of a specific job (JOB03361), enter:

```
ls -l /ftadv:filetype=JES,C=ISO8859-1,D=IBM-1047/JOB03361/
```

Or, alternatively:

```
cd /ftadv:filetype=JES,C=ISO8859-1,D=IBM-1047/JOB03361/
ls -l
```

You should see this type of output:

<table>
<thead>
<tr>
<th>JOB03361</th>
<th>USER1</th>
<th>USERID</th>
<th>A</th>
<th>J</th>
<th>0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0002</td>
<td>JES2</td>
<td>JESMSG</td>
<td>19</td>
<td>1100</td>
<td>UA</td>
</tr>
<tr>
<td>0003</td>
<td>JES2</td>
<td>JESJCL</td>
<td>8</td>
<td>390</td>
<td>V</td>
</tr>
<tr>
<td>0004</td>
<td>JES2</td>
<td>JESYSMSG</td>
<td>15</td>
<td>817</td>
<td>VA</td>
</tr>
<tr>
<td>0102</td>
<td>S1</td>
<td>SYSPRINT</td>
<td>5</td>
<td>416</td>
<td>VBA</td>
</tr>
</tbody>
</table>

**5.4 Securing the Client**

By default, the configuration of Tectia client tools for z/OS is aimed toward usability. If you want to have a minimal, high-security configuration, you have to disable some functionality.

Connecting to servers whose administrator you do not trust and who may be malicious is a significant security risk. Avoid it. However, there are a few ways to reduce the risk.

**5.4.1 Disabling Agent Forwarding**

By default, when you have an agent running, the agent connection will be forwarded to the server side. This is for convenience, so that a system administrator (or a power user) can easily connect from machine to machine.
If the server's administrator is malicious, he can use your agent to sign requests. This will not allow access to the private key, but will cause a security risk, because the malicious admin can then connect with your credentials anywhere where they are accepted.

If you do not trust the server, disable agent forwarding in the `ssh-broker-config.xml` file (either under `default-settings` or per each connection profile):

```xml
<forwards>
  <forward type="agent" state="off" />
  ...
</forwards>
```

On the command line, the `-a` option has the same effect:

```
$ sshg3 -a host3.example.com
```

See `forwards` for more information.
Tectia client tools for 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.

For easier migration, Tectia client tools for z/OS provide automatic FTP-SFTP conversion and transparent FTP tunneling. In these cases, no changes to the existing FTP client applications and scripts are required, and they can remain being used as before.

This chapter deals with secure file transfer using the SFTP command-line tools. For information on FTP-SFTP conversion and transparent FTP tunneling, see Tectia Server for IBM z/OS Administrator Manual.

6.1 Native z/OS FTP commands versus Tectia SFTP commands

Tectia client tools for z/OS convert the unsecured FTP subcommands to secure SFTP commands according to the following table:
### Table 6.1. Comparison between native z/OS FTP subcommands and SFTP commands

<table>
<thead>
<tr>
<th>z/OS FTP subcommand</th>
<th>Tectia SFTP command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ascii</td>
<td>ascii</td>
<td>Change the data transfer type to ASCII</td>
</tr>
<tr>
<td>binary</td>
<td>binary</td>
<td>Change the data transfer type to image</td>
</tr>
<tr>
<td>cd directory</td>
<td>cd directory</td>
<td>Change the working directory or file group on the remote host:</td>
</tr>
<tr>
<td>cd /mydir/test</td>
<td>cd /mydir/test</td>
<td>- when the remote end is Unix</td>
</tr>
<tr>
<td></td>
<td>cd //</td>
<td>- when the remote end is mainframe, go to MVS-level prefix.*</td>
</tr>
<tr>
<td>cd 'my.dataset.cntl'</td>
<td>cd //my.dataset.cntl</td>
<td>Define an absolute data set</td>
</tr>
<tr>
<td>cd my.dataset.cntl</td>
<td>cd //my.dataset.cntl</td>
<td>Define a relative data set</td>
</tr>
<tr>
<td>delete foreign_file</td>
<td>rm [options...] foreign_file</td>
<td>Delete the foreign_file (or a directory with the rm command)</td>
</tr>
<tr>
<td>delete your_file.txt</td>
<td>rm your_file.txt</td>
<td>Delete a file</td>
</tr>
<tr>
<td></td>
<td>rm yourdir</td>
<td>Delete a directory</td>
</tr>
<tr>
<td>dir name (DISK)</td>
<td>ls</td>
<td>View directory contents. See the options in the ls command</td>
</tr>
<tr>
<td>lcd directory</td>
<td>lcd directory</td>
<td>Change the current working directory on the local host</td>
</tr>
<tr>
<td>lcd 'my.dataset.cntl'</td>
<td>cd //my.dataset.cntl</td>
<td>Define an absolute data set</td>
</tr>
<tr>
<td>lcd my.dataset.cntl</td>
<td>cd //my.dataset.cntl</td>
<td>Define a relative data set</td>
</tr>
<tr>
<td>lmkdir directory (LIKE remote_directory)</td>
<td>lmkdir directory</td>
<td>Make a directory on the remote host</td>
</tr>
<tr>
<td>lmkdir example</td>
<td>lmkdir example</td>
<td>When the local dir is MVSUSER.FTP, then MVSUSER.FTP.EXAMPLE is created</td>
</tr>
<tr>
<td>locsite parameters...</td>
<td>ls site parameters...</td>
<td>Define parameters for controlling the file transfer. Many of the parameters are the same. See the manual.</td>
</tr>
<tr>
<td>ls name (DISK)</td>
<td>ls [-R] [-l] [-S] [-r] [-p] [-z]+z] [file...]</td>
<td>List only the names of a set of remote files, file group, or directory</td>
</tr>
<tr>
<td>ls bart</td>
<td>ls bart</td>
<td>List all MVS data sets with prefix USER1.TEST.PS.</td>
</tr>
<tr>
<td></td>
<td>ls //USER1.TEST.PS.</td>
<td>List all MVS data sets with prefix USER1.TEST.PS. (for a client that does not send the / character)</td>
</tr>
<tr>
<td>ls bart (DISK)</td>
<td>ls -l</td>
<td>Extended list command</td>
</tr>
<tr>
<td>get srcfile dstfile(PLACE</td>
<td>sget [options...] srcfile [dstfile]</td>
<td>SFTP has no option to store the output</td>
</tr>
<tr>
<td>z/OS FTP subcommand</td>
<td>Tectia SFTP command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>get myfile</td>
<td>sget myfile</td>
<td>Copy multiple files from a remote host to your local host, and create a corresponding number of local files. For SFTP 'replace' is the default!</td>
</tr>
<tr>
<td>mget foreign_file (REPLACE)</td>
<td>mget [options...] file</td>
<td>Note: The defaults are different!</td>
</tr>
<tr>
<td>mget foreign_file</td>
<td>mget --overwrite=no foreign_file</td>
<td></td>
</tr>
<tr>
<td>mkdir directory (LIKE local_directory)</td>
<td>mkdir directory</td>
<td>Create a directory on the remote host</td>
</tr>
<tr>
<td>mkdir 'MY.TEST.PDS'</td>
<td>mkdir '//MY.TEST.PDS'</td>
<td>Create a partitioned data set on the remote host</td>
</tr>
<tr>
<td>mput local_data_set (REPLACE)</td>
<td>mput [options...] file</td>
<td>Copy multiple data sets from your local host to the remote host. For SFTP 'replace' is the default!</td>
</tr>
<tr>
<td>mput foreign_file</td>
<td>mput --overwrite=no foreign_file</td>
<td>Note: The defaults are different!</td>
</tr>
<tr>
<td>open hostname portnumber</td>
<td>open hostname ] -l</td>
<td>Open a connection to the given host, or to local host: open -l</td>
</tr>
<tr>
<td>open remhost</td>
<td>open REMUSER@remhost</td>
<td>Open a connection to a remote host</td>
</tr>
<tr>
<td>put srcfile dstfile(REPLACE)</td>
<td>sput [options...] srcfile [dstfile]</td>
<td>The mainframe data set specification syntax can be different per client</td>
</tr>
<tr>
<td>put 'myfile.dataset.cnt' c:\bart.txt</td>
<td>sput '//myfile.dataset.cnt' c:\bart.txt</td>
<td>When advice is used, the preferred data set specification is different</td>
</tr>
<tr>
<td>rename oldfile newfile</td>
<td>rename oldfile newfile</td>
<td>Rename a file</td>
</tr>
</tbody>
</table>

**IBM z/OS:** For MVS data sets, if the data set that is specified by the *newfile* value already exists, the server rejects the rename request.

For z/OS UNIX files and named pipes, if the file that is specified by the *newfile* value already exists, the existing file is replaced.

**Tectia:** Tries to rename the *oldfile* to *newfile*. If the new file already exists, the files are left intact.

| sendsite | not applicable | Toggle the automatic sending of the SITE commands when sending a data set to a foreign host. |
| quit     | quit           | |
| bye      | bye            | The same as quit. |
6.2 Secure File Transfer with scpg3 and sftpg3 Commands

Tectia client tools for z/OS provide commands **scpg3** (secure copy) and **sftpg3** for secure file transfer. These command-line clients apply the Secure File Transfer Protocol (SFTP).

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

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

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

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

6.2.1 Using scpg3

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

The basic syntax of **scpg3** is:

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

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

Tectia Server for IBM z/OS uses the user's Unix System Services (USS) home directory as the default file transfer home location. The environment variable **SSH_SFTP_HOME_MVS** in the user's **$HOME/.ssh2/environment** file on the server can be used to control this location. For more information, see Section 6.4.2.

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

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

For more information on the command-line options, see **scpg3(1)**.
6.2.2 Using sftpg3

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

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

The basic syntax of sftpg3 is:

```
sftpg3 user@host
```

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

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

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

6.2.3 Enhanced File Transfer Functions

The following enhanced file transfer features are available with FTP-SFTP conversion and with the scpg3 and sftpg3 command-line tools of Tectia client tools for z/OS:

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

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

6.3 Handling MVS Data Sets and HFS File System Access

This section describes how access to MVS data sets and Hierarchical File System (HFS) files is handled with Tectia client tools for z/OS.
6.3.1 Data Set and HFS File System Access

z/OS has both MVS data sets, and HFS files. As both types must be accessed by the SFTP server, there must be a mechanism for distinguishing between them. Traditionally, MVS data sets in z/OS are accessed using the file name format "//'NAME.OF.MVS.DATASET'", while HFS files are accessed using the file name format /path/to/hfs/file.

In z/OS, if a data set name is not enclosed in single quotes, the user prefix is added in front of the data set name. For example, if user USER1 has a data set DATASET.NAME1, the user can access it using the data set name //DATASET.NAME1. It is also possible to use an absolute prefixed name "//'USER1.DATASET.NAME1'".

z/OS has also library data sets, whose members are accessed using the data set name "//DATASET.NAME1(MEMBER1)".

Note that the MVS data set names mentioned above are placed within regular quotation marks (" "). This must be done in shell commands (for example when using scpg3), to prevent the single quotes and parentheses in the data set names from being interpreted by the shell. Alternatively, you can use backslashes (\) to escape the single quotes and parentheses, for example ///\'USER1.DATASET.NAME1\' or ///DATASET.NAME1\(MEMBER1\).

System symbols can be used in data set names and volume serial numbers. The symbols are resolved on the host where the data set resides.

Case-Sensitivity of HFS and MVS Names

HFS file names are case-sensitive. For example, /tmp/MYFILE and /tmp/myfile result in two different files.

MVS data set names are case-insensitive. For example //USER1.DATASET.NAME1' and //user1.dataset.name1' are handled the same way.

6.3.2 Data Set Access Using DD Cards

When running the client programs scpg3 and sftpg3 in JCL, the _BPX_SHAREAS environment variable must be set to no. Local data sets can be allocated by using DD cards. For the DD cards to work, BPXBATS must be used instead of BPXBAT. BPXBATS uses local spawn and forces scpg3 and sftpg3 to run in the original address space where the DD information is available.

The following sftpg3 commands cannot be used with DD names:

- rm (remove file)
- rename
- mkdir
- rmdir
Empty data sets cannot be read when referred to by DD names.

For correct syntax, see the SCPPUT and SCPGET2 examples in Section 6.6.2.

### 6.3.3 Accessing Generation Data Groups (GDG)

Tectia Server for IBM z/OS supports generation data groups defined in ICF. Reading GDG ALLs is not supported. Tectia Server for IBM z/OS will not create GDG bases or model data sets.

**Note**

Tectia file transfers are atomic. Running "sget gdg.base(0) file1" two times in succession might retrieve different files. A loop of "sput file1 gdg.base(+1)" commands might fill the GDG with identical files and roll off all the previous generations.

**Navigating**

`sftpg3` allows you to navigate to a GDG base and to a prefix of a base.

**Listing**

Generation data sets (GDSs) are normal data sets. The long name format (`ls -l`) will show all details.

Listing a base with `-l` will show full details of the GDSs. The listing may contain data sets that are not in the GDG index but do have data set names that have the GDG name as a prefix.

It is possible but not recommended to use data set names which have the GDG base name as a prefix but are not GDS names. For example:

```bash
sftp> cd //USER1.GENGPR
MVS prefix `''USER1.GENGPR.'' is the current directory.
The working directory `''USER1.GENGPR.'' is a generation data group.
''USER1.GENGPR.''
sftp> ls -l
```

<table>
<thead>
<tr>
<th>Volume</th>
<th>Referred</th>
<th>Recfm</th>
<th>Lrecl</th>
<th>Blksz</th>
<th>Dsorg</th>
<th>Space</th>
<th>Dsname</th>
</tr>
</thead>
<tbody>
<tr>
<td>S6SYS1</td>
<td>Jan 03</td>
<td>VB</td>
<td>1000</td>
<td>27998</td>
<td>PS</td>
<td>50001</td>
<td>G0006V00.IMPOSTOR</td>
</tr>
<tr>
<td>S6SYS1</td>
<td>Jan 02</td>
<td>VB</td>
<td>1000</td>
<td>27998</td>
<td>PS</td>
<td>50001</td>
<td>G0007V00</td>
</tr>
<tr>
<td>S6SYS1</td>
<td>Jan 02</td>
<td>VB</td>
<td>1024</td>
<td>27998</td>
<td>PS</td>
<td>50001</td>
<td>G0008V09</td>
</tr>
<tr>
<td>S6SYS1</td>
<td>Jan 02</td>
<td>VB</td>
<td>1024</td>
<td>27998</td>
<td>PS</td>
<td>50001</td>
<td>G0077V99</td>
</tr>
<tr>
<td>S6SYS1</td>
<td>Jan 02</td>
<td>FB</td>
<td>80</td>
<td>27920</td>
<td>PS</td>
<td>50001</td>
<td>G0100V00</td>
</tr>
</tbody>
</table>

You cannot navigate to a prefix that ends in `GnnnnVnn` qualifier. Thus you cannot do "cd G0006V00" or "ls //''USER1.GENGPR.G0006V00''" in the example above.

If the GDG has the `NOSCRATCH` option, GDSs are retained when they are rolled off. `sftpg3` shows data sets based on the prefix - it does not show which data sets are in the GDG and which are not.
Access by Relative Data Set Name

`scp3` and `sftp3` give full access with relative generation numbers for reading and writing. For example, to read the previous and the latest generation, and create a new generation, do the following:

```
sftp> cd //'ABC.XYZ'
sftp> sget '-1' /tmp/yesterday
sftp> sget 0 /tmp/current
sftp> sput /tmp/new '+1'
```

z/OS may require you to specify a model data set when creating a new GDS. Tectia does not support the DCB=dsn specification, but you can use the LIKE attribute. Specify it with a file transfer advice string:

```
sftp> sput /tmp/new /ftadv:LIKE=USER1.GENGRPM.MODEL/'+1'
```

Alternatively, you can specify it with the `site` command (in `sftp3`):

```
sftp> site LIKE=USER1.GENGRPM.MODEL
sftp> sput /tmp/new '+1'
```

**Note**

GDG ALL is not supported (that is, reading all the generations as a concatenation).

A new GDS is rolled in immediately. You can not read it back as (+1) (you can do this in JCL, where the GDS generations are rolled in at the end of the job).

GDSs can be removed and renamed by the relative GDS name. On a rename operation, the new name must not be a relative GDS name.

Access by Absolute Data Set Name

With absolute GDS names you can do all the things possible with other data sets.

Writing a data set with a last qualifier with the GnnnnVnn format requires that there exists a suitable GDG base. If the generation exists it is overwritten. If it does not, the new file is inserted in its place in the GDG and older GDGs are rolled off, if necessary.

6.3.4 Accessing Migrated Data Sets

Tectia Server for IBM z/OS can transfer files to and from cataloged, online DASD (direct access storage device) data sets. The following data set types are supported:

- uncataloged DASD data sets
- migrated DASD data sets
- offline DASD data sets
• cataloged tape data sets
• uncataloged tape data sets
• offline tape data sets

**Tape Security**

Tectia relies on the system security product for controlling access to data sets. Sites should ensure that the appropriate restrictions are in place for **NL** (No Label) or **BLP** (Bypass Label Processing). Refer to the *RACF Security Administrator's Guide* or appropriate documentation for your SAF for information on restricting access to resource ICHBLP in the FACILITY class, or equivalent functionality. Tectia does not read labels and does not have any label exits.

**Operational Security and Tectia Controls**

Using an offline tape data set requires that a tape drive is available and that the tape cassette is mounted on the drive. The operator must mount the drive or cancel the mount request. Outstanding mount requests can cause shortages, locking and deadlocks in resource handling in the system.

Using data sets on offline DASD volumes requires that the operator varies the volume device online. This requires the operator to decide whether the device can be varied online (it is probably offline for a good reason) and to enter the vary command or cancel the request.

Using a migrated data set requires the system to recall the data set to a user-accessible (level 0) volume before it can be read or overwritten. If the data set is on a migration level 1 (DASD) or migration level 2 (tape) volume, it is first copied to level 0.

The **z/OS** system operators must monitor mount requests to make sure they do not go unanswered or cause problems. They must also monitor the system for Tectia SFT Server processes and Tectia file transfer clients that are hanging on allocation requests for offline data sets.

In Tectia Server for IBM z/OS, mounting an offline data set is by default not allowed and recalling a migrated data set is by default allowed.

The Tectia end user controls mounting and recalling with the extended file attributes **automount** and **autorecall**. The user must set **automount** to *yes* or *immed* for Tectia to attempt to mount offline data sets. The user can set **autorecall** to *no* to avoid recalling a data set by mistake.

In addition, the administrator must set up access rights to the RACF facility **SSZ.MOUNT** for the users who will be mounting offline data sets with Tectia client tools for **z/OS**. For more information, see *Tectia Server for IBM z/OS Administrator Manual*.

**Using Uncataloged Data Sets**

Existing data sets that do not have entries in the catalog can be accessed by specifying a volume serial number and unit.
Example 1. Read a tape data set that has the data set name `SRVACC1.SPECS.PDF` and is the third data set on volume 456123.

Specify the file transfer attributes (`UNIT, VOLUMES, DATASET_SEQUENCE_NUMBER`) using a file transfer advice string:

```
sftp> sget /ftadv:un=TAPE,vol=456123,seqnum=3///'SRVACC1.SPECS.PDF' \
      /home/clusrl/tmp/specs.pdf
```

Alternatively, you can use the `sftpg3 site` command:

```
sftp> site unit=TAPE,volumes=456123,dataset_sequence_number=3 

sftp> sget //'SRVACC1.SPECS.PDF' /home/clusrl/tmp/specs.pdf
```

Example 2. Create a data set without cataloging it using the `NORMDISP` attribute.

Specify the file transfer attributes (`UNIT, NORMDISP`) using a file transfer advice string:

```
sftp> sput /home/clusrl/tmp/specs-v2.pdf \
      /ftadv:unit=TAPE,normdisp=KEEP///'SRVACC1.SPECS.V2.PDF'
```

Alternatively, you can use the `sftpg3 site` command:

```
sftp> site unit=TAPE,normdisp=KEEP 

sftp> sput /home/clusrl/tmp/specs-v2.pdf //'SRVACC1.SPECS.V2.PDF'
```

The `DATACLAS, STORCLAS,` and `MGMTCLAS` attributes are also available.

**Requesting the System to Mount a Volume**

For Tectia to request the system to mount an offline volume, you must use the `AUTOMOUNT` attribute. You must also have `READ` access to `SSZ.MOUNT`. For example, to read a data set on a tape volume, assuming that the data set has a catalog entry, use the following commands.

Specify the attribute with a file transfer advice string:

```
sftp> sget /ftadv:automount=yes///'SRVACC1.SPECS.PDF' \
      /home/clusrl/tmp/specs.pdf
```

Alternatively, you can use the `sftpg3 site` command:

```
sftp> site automount=yes 

sftp> sget //'SRVACC1.SPECS.PDF' /home/clusrl/tmp/specs.pdf
```

When `automount=yes`, Tectia first attempts to allocate the data set without allowing the system to mount the data set if it is offline. If this allocation fails, Tectia retries the allocation with the mount request. You can make Tectia omit the first attempt by specifying `automount=immed`. 
Low-Level Access

Tectia has a facility for low-level access to dynamic file allocation because of the large number of attributes that can be specified when allocating tape data sets.

Tectia allocates data sets by calling the SVC 99 interface. This interface has a table of text units. Each text unit consists of key, parameter count, and parameters. The parameters have a length field and data. The interface is documented in the *Authorized Assembler Services Guide*.

Tectia allocates files at the start of a file transfer (except if the file is already allocated and is represented by a DD name). Tectia sets text units based on the extended attributes. The FTADV/site parameter *svc99_text_units* allows the user to add and change text units in the table. The value of *svc99_text_units* is a sequence of text units delimited by underscores (‘_’). Each text unit has the format *xxxxFvalues*, where *xxxx* is the key in hexadecimal, *F* is a format character and *values* is a character string. The *values* part is empty or consists of one string or of several strings delimited by plus signs (‘+’). No value may contain spaces, tabs, newlines or any of the characters ‘_+,/\’. The *values* part must not exceed 80 characters in length.

The format character indicates how the value(s) are encoded. The possible values are:

- **S**: character string
- **X**: hexadecimal
- **F**: decimal number with length 4 in the text unit
- **H**: decimal number with length 2 in the text unit
- **C**: decimal number with length 1 in the text unit
- **-**: no text unit

When the format character is ‘-‘, Tectia removes the text unit with the given key if it is in the table. For the other characters, Tectia replaces the value in the text unit with the given key or adds a new text unit.

The following attributes show how to encode some of the examples in the *Authorized Assembler Services Guide*. The attributes are not intended to be practical cases.

To specify an accessibility code of Z for an ANSI tape data set:

- Using a file transfer advice string:

  ```
sftp> cd /ftadv:svc99_text_units=8001XE9/
  ```

- Using the **site** command:

  ```
sftp> site svc99_text_units=8001XE9
  ```

To specify a RACF profile that was defined generically and request unallocation when a DCB is closed:
• Using a file transfer advice string:
  sftp> cd /ftadv:svc99_text_units=800EXD9D7D9D6C6+80_001CX/

• Using the `site` command:
  sftp> site svc99_text_units=800EXD9D7D9D6C6+80_001CX

To enter a Key Encode Specification and a Key Label:

• Using a file transfer advice string:
  sftp> cd /ftadv:svc99_text_units=8026SH_8024SLABELQ1.LABELQ2.LABELQ3/

• Using the `site` command:
  sftp> site svc99_text_units=8026SH_8024SLABELQ1.LABELQ2.LABELQ3

6.3.5 SFTP and Tape Data Sets

When working with tape data sets, there are two alternative ways to specify the device unit on which the tape is to be mounted. Normally this can be done through the use of the FTADV/site parameter `UNIT`, but in some situations it is not practical to do so. The environment variable `SSH_SFTP_VOL_UNIT_MAP` can be defined to provide a mapping between a volume serial number pattern and a device unit, allowing the unit to be deduced from the volume.

`SSH_SFTP_VOL_UNIT_MAP` may be set to a picture-clause type pattern matching a six character volume serial number, and followed by the unit to be used, in the form `SSH_SFTP_VOL_UNIT_MAP=volume_pattern:unit_specification`.

For example, `SSH_SFTP_VOL_UNIT_MAP=AA9999:CART`, where `A` is an alphabetic character and `9` is a decimal digit, means that volumes with a serial number consisting of two letters followed by four digits are to be allocated with `UNIT=CART`.

More specifically, `SSH_SFTP_VOL_UNIT_MAP=TP00001:CART` generates an allocation request for a tape with volume serial number `TP0001` with unit `CART`.

6.4 Controlling File Transfer

The current Secure File Transfer Protocol (SFTP) does not transfer any information about the files to be transferred, only the file contents as a byte stream. This is sufficient for Unix-type files if the sender and receiver use the same CCS. However, it is possible to set specific file permissions for transferred files by using the `chmod` command.
With MVS data sets, Tectia needs more information: which transfer format to use, what coded character sets are involved, and what the file characteristics are. The information can be encoded in the file name with a file transfer advice string, or it can be read from a file transfer profile. When both the client and server are Tectia, alternatively the site commands of \texttt{s cp g 3} and \texttt{sftp g 3} can be used.

File transfer advice string and site command are described in the following section. For more information on file transfer profiles, see Section 9.4.2.

6.4.1 File Transfer Advice String / Site Command

File transfer advice strings (FTADV) can be used to relay the information needed in the file transfer by encoding it in the file name. Any file name that starts with \texttt{/ftadv:} has the format

\begin{verbatim}
/ftadv:advstr/realfilename
\end{verbatim}

where \texttt{advstr} is the advice string and \texttt{realfilename} is the data set name or a file name to be further processed by the server.

The advice string is either a sequence of name=value pairs delimited by commas, or a one-word shortcut. For ease of use, some advice string names also have abbreviations. For example: \texttt{/ftadv:automount=yes/}, \texttt{/ftadv:automount/} and \texttt{/ftadv:autom/} all enable automount.

When using Tectia file transfer clients, site commands can be used as an alternative for file transfer advice string. For site command descriptions, see the site and lsit e command in \texttt{sftp g 3(1)} and the \texttt{--dst-site} and \texttt{--src-site} options in \texttt{scpg 3(1)}.

Similarly to file transfer advice string, when giving the site command, either the full parameter name or its abbreviation can be used. For example, the following two commands accomplish the same thing:

\begin{verbatim}
sftp> site x=bin
sftp> site transfer_mode=bin
\end{verbatim}

The available FTADV/site parameters are listed and described in the following.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Abbreviations</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMOUNT</td>
<td>-</td>
<td>YES</td>
</tr>
<tr>
<td>[NO]AUTOMOUNT</td>
<td>[NO]AUTOM</td>
<td>-</td>
</tr>
<tr>
<td>AUTORECALL</td>
<td>-</td>
<td>YES</td>
</tr>
<tr>
<td>[NO]AUTORECALL</td>
<td>[NO]AUTOR</td>
<td>-</td>
</tr>
<tr>
<td>BLKSIZE</td>
<td>B, BLOCKS</td>
<td>size</td>
</tr>
<tr>
<td>BLOCKS</td>
<td>BL</td>
<td>-</td>
</tr>
<tr>
<td>CONDDISP</td>
<td>CO</td>
<td>CATLG</td>
</tr>
<tr>
<td>CYLINDERS</td>
<td>CY</td>
<td>-</td>
</tr>
<tr>
<td>DATACLAS</td>
<td>DA</td>
<td>class</td>
</tr>
<tr>
<td>DATASET_SEQUENCE_NUMBER</td>
<td>SEQNUM</td>
<td>number</td>
</tr>
<tr>
<td>DEFER</td>
<td>DE</td>
<td>YES</td>
</tr>
<tr>
<td>[NO]DEFER</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DIRECTORY_SIZE</td>
<td>M, DI, DIRSZ</td>
<td>size</td>
</tr>
<tr>
<td>EXPIRY_DATE</td>
<td>EXPDT</td>
<td>yyyyddd</td>
</tr>
<tr>
<td>FILE_STATUS</td>
<td>STATUS</td>
<td>NEW</td>
</tr>
<tr>
<td>FILETYPE</td>
<td>FILET</td>
<td>SEQ</td>
</tr>
<tr>
<td>FIXRECFM</td>
<td>FI</td>
<td>length</td>
</tr>
<tr>
<td>JOB_ID</td>
<td>JESID</td>
<td>ID</td>
</tr>
<tr>
<td>JOB_OWNER</td>
<td>JESO</td>
<td>name</td>
</tr>
<tr>
<td>JOBNAME</td>
<td>JESJOB</td>
<td>name</td>
</tr>
<tr>
<td>KEYLEN</td>
<td>KEYL</td>
<td>length</td>
</tr>
<tr>
<td>KEYOFF</td>
<td>KEYO</td>
<td>offset</td>
</tr>
<tr>
<td>LABEL_TYPE</td>
<td>LABEL</td>
<td>NL</td>
</tr>
<tr>
<td>LIKE</td>
<td>-</td>
<td>like</td>
</tr>
<tr>
<td>LRECL</td>
<td>R, LR</td>
<td>length</td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>MG</td>
<td>class</td>
</tr>
<tr>
<td>NORMDISP</td>
<td>NOR</td>
<td>CATLG</td>
</tr>
<tr>
<td>PRIMARY_SPACE</td>
<td>PRI</td>
<td>space</td>
</tr>
<tr>
<td>PROFILE</td>
<td>P, PROF</td>
<td>profile</td>
</tr>
<tr>
<td>RECFM</td>
<td>O, REC</td>
<td>recfm</td>
</tr>
<tr>
<td>RECORD_TRUNCATE</td>
<td>U, TRUN</td>
<td>YES</td>
</tr>
<tr>
<td>[NO]TRUNCATE</td>
<td>[NO]TRU, [NO]TRUN</td>
<td>-</td>
</tr>
<tr>
<td>RETENTION_PERIOD</td>
<td>RET</td>
<td>days</td>
</tr>
<tr>
<td>SECONDARY_SPACE</td>
<td>SE, SEC</td>
<td>space</td>
</tr>
<tr>
<td>SIZE</td>
<td>L</td>
<td>size</td>
</tr>
<tr>
<td>SPACE_RELEASE</td>
<td>RLSE</td>
<td>YES</td>
</tr>
<tr>
<td>SPACE_UNIT</td>
<td>SU</td>
<td>BLKS</td>
</tr>
<tr>
<td>Parameter</td>
<td>Abbreviations</td>
<td>Possible values</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>SPACE_UNIT_LENGTH</td>
<td>SUL</td>
<td>length</td>
</tr>
<tr>
<td>STAGING</td>
<td>S, STAGE</td>
<td>YES NO</td>
</tr>
<tr>
<td>STORCLAS</td>
<td>ST</td>
<td>class</td>
</tr>
<tr>
<td>SVC99_TEXT_UNITS</td>
<td>SVC99</td>
<td>string</td>
</tr>
<tr>
<td>TRACKS</td>
<td>TR</td>
<td>-</td>
</tr>
<tr>
<td>TRAILING_BLANKS</td>
<td>TRAIL</td>
<td>YES NO</td>
</tr>
<tr>
<td>[NO]TRAILING_BLANKS</td>
<td>[NO]TRAIL,</td>
<td>-</td>
</tr>
<tr>
<td>TRANSFER_CODESET</td>
<td>C, CODESET</td>
<td>codeset</td>
</tr>
<tr>
<td>TRANSFER_FILE_CODESET</td>
<td>D, FCODESET</td>
<td>codeset</td>
</tr>
<tr>
<td>TRANSFER_FILE_LINE_DELIMITER</td>
<td>J, FLDELIM</td>
<td>UNIX MVS MVS-FTP DOS MAC NEL</td>
</tr>
<tr>
<td>TRANSFER_FORMAT</td>
<td>F, FORMAT</td>
<td>LINE STREAM RECORD</td>
</tr>
<tr>
<td>TRANSFER_LINE_DELIMITER</td>
<td>I, LDELIM</td>
<td>UNIX MVS MVS-FTP DOS MAC NEL</td>
</tr>
<tr>
<td>TRANSFER_MODE</td>
<td>X, MODE</td>
<td>BIN TEXT</td>
</tr>
<tr>
<td>TRANSFER_TRANSLATE_DSN_TEMPLATES</td>
<td>A, XDSNT</td>
<td>templates</td>
</tr>
<tr>
<td>TRANSFER_TRANSLATE_TABLE</td>
<td>E, XTBBL</td>
<td>table</td>
</tr>
<tr>
<td>TYPE</td>
<td>T</td>
<td>PS PO PDS POE PDSE GDG HFS VSAM ESDS KSDS RRN</td>
</tr>
<tr>
<td>UNIT</td>
<td>UN</td>
<td>unit</td>
</tr>
<tr>
<td>UNIT_COUNT</td>
<td>UC, UNC</td>
<td>number</td>
</tr>
<tr>
<td>UNIT_PARALLEL</td>
<td>UNP</td>
<td>YES NO</td>
</tr>
<tr>
<td>VOLUME_COUNT</td>
<td>VC, VOLCNT</td>
<td>number</td>
</tr>
<tr>
<td>VOLUMES</td>
<td>VO, VGL</td>
<td>vol1+vol2+...</td>
</tr>
</tbody>
</table>

AUTOMOUNT=YES NO IMMED

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

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

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

Default: NO

[NO]AUTOMOUNT | [NO]AUTOM
AUTOMOUNT AUTOM is equal to AUTOMOUNT=YES.
NOAUTOMOUNT NOAUTOM is equal to AUTOMOUNT=NO.

AUTORECALL=YES NO

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

[NO]AUTORECALL | [NO]AUTOR

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

BLKSIZE | B | BLOCKSIZE=size

Specifies the maximum block size.

Default: none

BLOCKS | BL

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

CONDDISP | CO=CATLG | UNCATLG | KEEP | DELETE

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

Note

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

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

• CATLG: an MVS data set is retained and its name is cataloged. An HFS file is retained.

• UNCATLG: the name of an MVS data set is removed from the catalog but the data set is retained. An HFS file is retained.

• KEEP: an MVS data set is retained (if cataloged it will be still cataloged, if uncataloged it will be still uncataloged). An HFS file is retained.

• DELETE: the name of an MVS data set is removed from the catalog and the space allocated for the data set is released.

Default: CATLG

CYLINDERS | CY

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

DATACLASS | DA=class

Specifies the data class of a data set.

Default: none

DATASETSEQUENCENUMBER | SEQUENumber

Identifies the relative position of a data set on a tape volume.
**DEFER|DE-YES|NO**

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

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

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

Default: NO

[NO]DEFER|DE

DEFER|DE is equal to DEFER=YES.

NODEFER is equal to DEFER=NO.

**DIRECTORY_SIZE|M|DI|DIRSZ-size**

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

Default: 10

**EXPIRY_DATE|EXPDT=yyddd|yyyyddd**

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

Default: System default

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

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

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

**FILETYPE|FILET=SEQ|JES**

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

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

Default: SEQ
FIXRECFM|FI-length
   The data set organization is set to FB and the fixed record length is set to length.
   Default: none

JOB_ID|JESID-ID
   When in FILETYPE=JES mode, JOB_ID specifies that commands accessing the JES spool, such as get, apply only to jobs with a job ID that matches the supplied ID.
   When interfacing with JES using a file transfer advice string, using ls -l in conjunction with a job ID permits listing the spool files for that job.
   Commands get, sget, and so on, with a job ID can be used to retrieve the spool files for a given job.

JOB_OWNER|JESO-name
   When in FILETYPE=JES mode, JOB_OWNER specifies that commands accessing the JES spool, such as ls, and get, and so on, apply only to jobs with owner matching the supplied name.
   Default: Current user

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

KEYLEN|KEYL-length
   Specifies the length in bytes of the keys used in the data set.
   Default: none

KEYOFF|KEYO-offset
   Specifies the key offset; the position of the first byte of the key in records of the specified VSAM data set.
   Default: none

LABEL_TYPE|LABEL=NL|SL|NSL|SUL|BLP|LTM|AL|AUL
   The type of the label for the data set. This attribute corresponds to the first value in the LABEL parameter of the JCL DD statement.

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

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

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

Default: none

LRECL|R|LR=length

Maximum record length or fixed record length.

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

MGMTCLAS|MG=class

Specifies the management class of a data set.

Default: none

NORMDISP|NOR=CATLG|UNCATLG|KEEP|DELETE

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

Default: CATLG

PRIMARY_SPACE|PRI=space

Primary space allocation for a data set.

Default: none

PROFILE|P|PROF=profile

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

Default: none

RECFM|O|REC=rcfm

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

<table>
<thead>
<tr>
<th>Letter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Fixed</td>
</tr>
<tr>
<td>V</td>
<td>Variable</td>
</tr>
<tr>
<td>U</td>
<td>Undefined</td>
</tr>
<tr>
<td>B</td>
<td>Blocked</td>
</tr>
<tr>
<td>S</td>
<td>Spanned or standard</td>
</tr>
<tr>
<td>M</td>
<td>Machine line printer codes</td>
</tr>
<tr>
<td>A</td>
<td>ASA line printer codes</td>
</tr>
</tbody>
</table>

© 2007–2020 SSH Communications Security Corporation
If binary mode transfer is requested for reading recfm=V(B)(S) dataset with recfm=U specified in the sget request, the block meta information (BDW + RDW) will be transferred to the destination. After the read operation completes, the catalog and dcb information for the dataset will be maintained.

**RECORD_TRUNCATE | U | TRUN=|YES|NO**

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

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

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

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

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>F and FB</td>
<td>LRECL</td>
</tr>
<tr>
<td>V and VB</td>
<td>LRECL - 4</td>
</tr>
<tr>
<td>U</td>
<td>BLKSIZE</td>
</tr>
<tr>
<td>VSAM</td>
<td>MAXRECLEN</td>
</tr>
</tbody>
</table>

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

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

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

**Note**

When Tectia client tools for z/OS writes a data set with RECORD_TRUNCATE=YES, data loss may occur.
[NO]TRUNCATE | [NO]TRU | [NO]TRUN
TRUNCATE | TRU | TRUN is equal to RECORD_TRUNCATE=YES.
[NO]TRUNCATE | [NO]TRU | [NO]TRUN is equal to RECORD_TRUNCATE=NO.

RETENTION_PERIOD | RET=days
The retention period in days. After the retention period, the data set expires and the operating system can delete or overwrite the data set.
Default: System default

SECONDARY_SPACE | SE | SEC=space
Secondary space allocation for a data set.
Default: none

SIZE | L=size
Size estimate (in bytes) for data set allocation.
Default: 1000000

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

SPACE_UNIT | SU=BLKS | TRKS | CYLS | AVGRECLEN
Unit of space allocation for a data set.
Possible values for the space allocation unit are:
• BLKS: Blocks
• CYLS: Cylinders
• TRKS: Tracks
• AVGRECLEN: Average record length
Default: none

SPACE_UNIT_LENGTH | SUL=length
When SPACE_UNIT=BLKS or SPACE_UNIT=AVGRECLEN, specifies the size of the space allocation unit.
Default: 100 with SPACE_UNIT=AVGRECLEN, none with SPACE_UNIT=BLKS

STAGING | S | STAGE=YES | NO
Specifies whether staging is to be used in the SFTP server when accessing a file or data set.
If set to **NO**, staging is not used.

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

Default: **NO**

**Note**

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

**STORCLAS | ST=class**

Specifies the storage class of system managed storage.

Default: none

**SVC99_TEXT_UNITS | SVC99=string**

Dynamic allocation arguments that override or are added to arguments from other file transfer attributes. For detailed information on this attribute, see the section called “Low-Level Access”.

Default: none

**TRACKS | TR**

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

**TRAILING_BLANKS | TRAIL=YES | NO**

This option specifies whether to preserve trailing blanks in a transferred fixed record (RECFM=F|FB) data set. This option only applies to line-delimited target files (TRANSFER_FORMAT=LINE), not to target unit-record data sets.

**Note**

In variable format datasets (RECFM-V|VB) trailing blanks are always preserved, independent of the value of this option.

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

If set to **NO**, trailing blanks will be stripped.

Default: **NO**

**[NO]TRAILINGBLANKS | [NO]TRAIL**

TRAILINGBLANKS | TRAIL is equal to TRAILING_BLANKS=YES.

NOTRAILINGBLANKS | NOTRAIL | NOTAIL is equal to TRAILING_BLANKS=NO.
TRANSFER_CODESET | CODESET=codeset

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

```
> iconv -l
```

Default: none

**Example 1: A Windows SFTP client puts a file to a z/OS data set and gets a data set from z/OS**

Using file transfer advice strings:

```
sftp> sput file.txt /ftadv:C=ISO8859-1,D=IBM-1047/__DATASET.TXT
sftp> sget /ftadv:D=IBM-1047,C=ISO8859-1/__DATASET.TXT file.txt
```

1. The Windows client tells the z/OS server that the code set during the transfer is ISO8859-1 and that the server should store the data set with the IBM-1047 code set.
2. The Windows client tells the z/OS server that the code set in the data set is IBM-1047 and it should be converted to ISO8859-1 before transferring the data to the Windows client.

Using the `site` command:

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

**Example 2: A z/OS SFTP client puts a data set to a Windows file and gets a file from Windows**

Using file transfer advice strings:

```
sftp> sput file.txt /ftadv:C=ISO8859-1,D=IBM-1047/__DATASET.TXT
sftp> sget /ftadv:C=ISO8859-1,D=IBM-1047/__DATASET.TXT file.txt
```

1. The z/OS client is told that code set in the data set is IBM-1047 and it should be converted to ISO8859-1 before transferring the data to the Windows server.
2. The z/OS client is told that the file has the ISO8859-1 code set during the transfer and the data set should be stored with the IBM-1047 code set.

Using the `lsite` command:

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

**Note**

The line delimiter information is always given to the host that is capable of performing the conversion, in these cases the z/OS host.
The data in the data set has the specified code set. codeset is the code set name that is known to the iconv function of the system performing the conversion. The available code sets can be listed by invoking the iconv command at a USS prompt with the -1 option:

```
> iconv -l
```

Default: none

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

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

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

**Note**

If you use J=MVS-FTP when writing data sets, you must set --streaming=no.

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

You can specify code sets by defining TRANSFER_FILE_CODESET without TRANSFER_CODESET. For example, to have a DOS delimiter in Unicode (x'00D00A0') appended to the records, set "I=DOS, J=MVS-FTP, C=UCS-2", and to have a Unix delimiter in ISO Latin 1 (x'0A0'), set "I=UNIX, J=MVS-FTP, D=ISO8859-1".

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

Default: none
Note

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

Note

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

Note

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

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

In this example, the code set is also converted.

Using file transfer advice strings:

```
sftp> sput /ftadv:I=DOS,J=MVS,C=ISO8859-1,D=IBM-1047/__DATASET.TXT file.txt
sftp> sget file.txt /ftadv:I=DOS,J=MVS,C=ISO8859-1,D=IBM-1047/__DATASET.TXT file.txt
```

1. The file line delimiter conversion (`J=MVS`) makes the z/OS client insert an NL (0x15) character after each record. The line delimiter conversion (`I=DOS`) converts all NL:s to CRLF (0x0D 0x0A) characters, which remain unchanged in the code set conversion. The z/OS client is told that the code set in the data set is IBM-1047 and it should be converted to ISO8859-1 before transferring the data to the Windows server.

2. The CRLF line delimiters (set by `I=DOS`) are converted to LF characters (`J=MVS`), which are converted to NL characters in the code set conversion. The conversion from line-delimited stream format to record-oriented format occurs automatically. In the code set conversion, the z/OS client is told that the file has the ISO8859-1 code set during the transfer and the data set should be stored with the IBM-1047 code set.

Using `lsite` commands:

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

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

The byte stream consists of the bytes that are transferred as payload in the SFTP protocol packets. The byte stream has one of the following formats: `LINE`, `STREAM` or `RECORD`. All three formats may have data consisting of text, non-text data, or a mixture of these.
When writing an MVS data set, a record that is longer than the maximum or fixed record length will cause an error unless RECORD_TRUNCATE is set to YES, in which case the record will be truncated. When writing to data sets with fixed record lengths, short records will be filled with binary zeroes if you use the record transfer format and with blanks if you use the line transfer format.

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

<table>
<thead>
<tr>
<th>Delimiter</th>
<th>EBCDIC</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Dec</td>
<td>Oct</td>
</tr>
<tr>
<td>\r Carriage Return</td>
<td>13</td>
<td>015</td>
</tr>
<tr>
<td>\n Newline</td>
<td>21</td>
<td>025</td>
</tr>
<tr>
<td>\f Form Feed</td>
<td>12</td>
<td>014</td>
</tr>
</tbody>
</table>

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

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

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

Tectia client tools for z/OS sends \n as the Server Newline Convention in the server initialization SFTP protocol message.

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

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

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

- **STREAM**: The stream transfer format contains the data bytes of the data set but no structural information. If a data set with a fixed record length is transferred with the stream format and recreated with the same record length, the record structure will be preserved. Variable length records will not be recreated properly if transferred with the stream format.
• **RECORD:** The record transfer format is record-based. Each record is preceded by a length field consisting of a 4 bytes, which indicates the number of data bytes in the record.

When `TRANSFER_FILE_LINE_DELIMITER | J=MVS` is specified, the length field consists of a 4-byte big-endian binary integer, which indicates the number of data bytes in the record.

When `TRANSFER_FILE_LINE_DELIMITER | J=MVS-FTP` is specified, the length field consists of a 2-byte big-endian binary integer followed by 2-byte zeros. The 2-byte big-endian binary integer indicates the number of data bytes in the record plus the four bytes length field.

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

Default: **LINE**.

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

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

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

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

**Note**

If you use `I=MVS-FTP` when reading data sets, you must set `--streaming=no`.

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

Default: none

**Note**

The line delimiter information should be given to the host that is capable of performing the conversion, such as a host with a Tectia.
TRANSFER_MODE | X | MODE=BIN | TEXT

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

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

Default: none

Note

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

TRANSFER_TRANSLATE_DSN_TEMPLATES | A | XDSNT=templates

*templates* specifies the search templates for the translate table. Write '%T' to show the point where the translate table name (see above) is to be inserted. Delimit the templates with a plus character. The data set name templates must not contain slashes, instead they must be preceded by two or three underscores. For more information, see Section 9.3.1.

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

Note

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

Default: none

TRANSFER_TRANSLATE_TABLE | E | XTB|table

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

TYPE | T=PS | PO | PDS | POE | PDSE | GDG | HFS | VSAM | ESDS | KS|DNS | RRR

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

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

• **GDG**: The type of the created data set is GDG.

• **HFS**: The type of the created data set is HFS.

• **VSAM**: The type of the created data set is VSAM.

• **ESDS**: The type of the created data set is VSAM ESDS.

• **KSDS**: The type of the created data set is VSAM KSDS.

• **RRN**: The type of the created data set is VSAM RRN.

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

**UNIT|UN=unit**

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

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

**Default:** none

**UNIT_COUNT|UC|UNC=number**

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

**Default:** System default

**UNIT_PARALLEL|UNP=YES|NO**

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

**Default:** System default

**VOLUME_COUNT|VC|VOLCNT=number**

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

**Default:** System default

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

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

**Default:** none
File Transfers between z/OS Machines

When data sets are transferred between z/OS machines, the destination data set is by default allocated with the same attributes as the source data set. The attributes for the destination data set can be overridden with the `site` commands.

**Note**

If you change the data set record format from `VB` to `FB` during transfer, you have to specify both `BLKSIZE` and `LRECL` for the destination data set. Otherwise, an error may occur if the block size does not match the record length.

**Note**

Tectia Server for IBM z/OS does not support data set allocation of `LARGE`, `EXTENDED`, or `COMPRESSED` formats. However, transfer is possible to pre-allocated data sets of these types.

### 6.4.2 File Transfer Environment Variables for the Clients

The file transfer clients `scp3` and `sftpg3` use the following environment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SSH_DEBUG_FMT</code></td>
<td>&quot;%Dd/%Dt/%Dy %Dm:%Ds:%Df %m/%s:%n:%f %M&quot;</td>
</tr>
<tr>
<td><code>SSH_SFTP_SMF_TYPE</code></td>
<td>NULL</td>
</tr>
<tr>
<td><code>SSH_SFTP_PSEUDOVOLUME_VOLSERS</code></td>
<td>MIGRAT</td>
</tr>
<tr>
<td><code>SSH_SFTP_HOME_MVS</code></td>
<td>yes</td>
</tr>
<tr>
<td><code>SSH_SFTP_SHOW_BYTE_COUNT</code></td>
<td>no</td>
</tr>
<tr>
<td><code>SSH_SFTP_STATISTICS</code></td>
<td>yes</td>
</tr>
</tbody>
</table>

The `SSH_DEBUG_FMT` variable can be used to specify the format of the debug messages. For more information, see `SSH_DEBUG_FMT`.

If `SSH_SFTP_SMF_TYPE` is set to `TYPE119`, file transfers create SMF records of type 119.

`SSH_SFTP_PSEUDOVOLUME_VOLSERS` can be used to define a list of pseudo-volume serial numbers for data sets that are migrated or archived, see Section 6.4.3.

The default file transfer home location is the user's Unix System Services (USS) home directory. If `SSH_SFTP_HOME_MVS` is set to `yes`, the local directory is set to USER prefix in the MVS side. See the section called “Setting the File Transfer Home Location” for examples of using this variable.

If `SSH_SFTP_SHOW_BYTE_COUNT` is set to `yes`, the number of transferred bytes is shown after successful file transfer. Also the names of source and destination files are shown.

By default, a normal progress bar is shown while transferring a file. If `SSH_SFTP_STATISTICS` is set to `no`, the progress bar is not shown. If it is set to `simple`, file transfer statistics are shown after the file has been transferred.
Furthermore, *sftpg3* uses the following environment variable for defining the path to the batch file to be run when *sftpg3* is started:

```
SSH_SFTP_BATCH_FILE   (default: NULL)
```

## Setting the File Transfer Home Location

For SFTP connections, the file transfer home location is the directory on the client where the SFTP session starts. By default, Tectia Server for IBM z/OS uses the user's Unix System Services (USS) home directory as the file transfer home location. The environment variable `SSH_SFTP_HOME_MVS` in the user's `$HOME/.profile` file on the client machine can be used to control the file transfer home location.

If `SSH_SFTP_HOME_MVS` is omitted or its value is `no`, the user's USS home directory is used as the file transfer home, for example `/u/userid/`, and the MVS user prefix must be accessed using "///".

If the value of `SSH_SFTP_HOME_MVS` is `yes`, the user's MVS USERID prefix is used as the file transfer home location, for example `// USERID.`, and the USS home directory must be accessed using `/u/userid/` or `~`.

### Examples when `SSH_SFTP_HOME_MVS=no`

When `SSH_SFTP_HOME_MVS` is set to `no` (or omitted), the following `get` command run in the SFTP client results to a file `/home/user1/dataset.txt` in Tectia Server for IBM z/OS:

```
sftp> open user1@server
sftp> get dataset.txt
```

The following `sget` command run in the SFTP client results to a MVS sequential data set `// USERID.MF.FILE` in Tectia Server for IBM z/OS:

```
sftp> open user1@server
sftp> sget remote_file //MF.FILE
```

### Examples when `SSH_SFTP_HOME_MVS=yes`

When `SSH_SFTP_HOME_MVS` is set to `yes`, the following `get` command in the SFTP client results to a data set `// USER1.DATASET.TXT` in Tectia Server for IBM z/OS:

```
sftp> open user1@server
sftp> get dataset.txt
```

The following `sget` command run in the SFTP client results to a USS file `/u/user1/mf.file` in Tectia Server for IBM z/OS:

```
sftp> open user1@server
sftp> sget remote_file ~/mf.file
```

Or:

```
sftp> open user1@server
sftp> sget remote_file /u/user1/mf.file
```
6.4.3 Restoring Archived Data Sets

A data set may have been migrated or archived by storing it out of the normal data file system to a tape or some other storage medium. Such data sets will have a volume serial number set to a pseudo-value that indicates that they are not available on an online volume. Data facility hierarchical storage manager (DFHSM) uses the special volume serial number `MIGRAT` for this purpose. The SFTP subsystem recognizes this by default and issues the appropriate allocation request to cause such data sets to be recalled when asked to do so.

Various other storage management systems use different values as the pseudo volume serial number for data sets they have migrated. You can give any such special volume serial number via the environment variable `SSH_SFT_PSEUDOVOLUME_VOLSERS`, which accepts a comma-separated list of one or more volume serial numbers, matching those in use by the local site storage manager to denote migrated data sets. In the case where migrated data sets at your installation have volume serial numbers other than `MIGRAT`, you can specify this environment variable in your shell profile or `STDENV DD`. See *Tectia Server for IBM z/OS Administrator Manual* for further details.

Extended file attributes `automount` and `autorecall` are used to control the behavior of SFTP when working with migrated data sets. See Section 6.4 for further details. Note that the requisite security manager permissions may be required to recall data sets.

6.5 Listing Data Sets with Tectia client tools for z/OS

Tectia client tools for z/OS can be used to list data sets.

6.5.1 Data Set Lists

To display data set lists, use the `ls` and `ls -l` commands. The `ls` command shows the list with relative data set names. The `ls -l` command shows additional information formatted by the server.

When listing a PDS or PDSE, the member names are listed.

When listing a GDG, the GDG name is treated as a prefix. The listing may contain data sets that are not in the GDG index but do have data set names that have the GDG name as a prefix.

Using the `ls` command without any parameters displays the content of the current working directory.
Users can also define the DSN qualifier or HFS path they want to list. To use an absolute DSN, start the prefix with "///". To use a relative DSN, which will be completed with the user name, start the prefix with "///".

Note

When listing data sets in the main catalog, the listing shows only catalog entries. The catalog does not have entries for users. The user's name will be in the listing only if there is a user catalog connector entry with that name.

To list data sets with the DSNs `USER1.TEST.PS.*`, use the following command:

```
sftp> ls //USER1.TEST.PS.'
//USER1.TEST.PS.:'
FILE1
FILE2
FILE3
```

If the user is connected as `USER1`, use the following command for an extended list:

```
sftp> ls -l //TEST.PS.
//TEST.PS.:
Volume Referred Recfm Lrecl BlkszDsorg Space Dsname
Z6SYS1 Jul 25 2006 VB  1024 27998 PS   50001  FILE1
Z6SYS1 Jul 25 2006 VB  1024 27998 PS   50001  FILE2
Z6SYS1 Jul 25 2006 VB  1024 27998 PS   50001  FILE3
```

The fields in the extended list are:

Volume
The volume or volumes the data set resides on. If the volume cannot be accessed, the text ":offline" is appended to the volume name.

Date (Referred)
The day on which the data set was last referred to, or the creation date

Record format (Recfm)
The `RECFM` or "VSAM"

Record size (Lrecl)
The `LRECL`

Block size (Blksz)
The `BLKSIZE`
File type (Dsorg)
The file type: PS, PO, POE, ESDS, KSDS, RRN, or PS/PO. PS/PO is used for non-VSAM data sets on off-line volumes.

Size Estimate (Space)
An estimate of the count of data bytes in the data set which is based on the number of used tracks; or for VSAM files, the High RBA; and for PDSEs, the number of allocated tracks. A 3390 track is estimated to hold 50001 bytes (the track count can be seen in the least significant digits). The size may be smaller than the number of data bytes in the data set, if it has an efficient block size. The size of the data set when transferred may differ greatly from the size estimate, depending on such factors as character encoding, line delimiters, and trailing blanks.

Name (Dsname)
The relative data set name

6.5.2 Data Set Hierarchy
Tectia client tools for z/OS present z/OS data sets as if they formed a hierarchy of directories where the data set name (DSN) qualifiers are the directory names.

The cd command can be used to change the DSN prefix. To use an absolute DSN start the prefix with "/". To use a relative DSN, which will be completed with the user name, start the prefix with "/".

For example, to change the working "directory" to "/USER1", use the command:

```
sftp> cd //USER1.
MVS prefix User1." is the current directory.
'USER1.'
sftp>
```

If the user is connected as USER1, use the command:

```
sftp> cd //
MVS prefix User1. is the current directory.
'USER1.'
sftp>
```

Once working in a "directory" (//USER1., for example) you can go into a "subdirectory" by doing a cd to a qualifier:

```
sftp> cd TEST.
MVS prefix User1.TEST." is the current directory.
'USER1.TEST.'
sftp> ls -l
'USER1.TEST.:'
Volume Referred Recfm Lrecl BlkSz Dsorg Space Dsname
Z6SYS1 Jul 17 2006 VB 1024 27998 PS 50001 FILE
Z6SYS1 Jul 25 2006 VB 1024 27998 PS 50001 PS.FILE1
Z6SYS1 Jul 25 2006 VB 1024 27998 PS 50001 PS.FILE2
```
You can go "up" in the hierarchy with the "cd .." command:

sftp> cd ..
MVS prefix "USER1.TEST." is the current directory.
'USER1.TEST.'
sftp> ls
'USER1.TEST.:'
FILE
PS.FILE1
PS.FILE2
PS.FILE3

sftp> cd ..
MVS prefix "USER1." is the current directory.
'USER1.'
sftp>

The cd command can also be used for going into a PDS or PDSE file, for example:

sftp> cd PDS
MVS prefix "USER1.PDS" is the current directory.
The working directory "USER1.PDS" is a partitioned data set.
'USER1.PDS'
sftp> ls
'USER1.PDS.:
MEM1
MEM2

Instead of using the cd command to change the directory one qualifier at a time, the whole data set name can be used, for example:

sftp> cd //"USER1.TEST.PS."
MVS prefix "USER1.TEST.PS." is the current directory.
'USER1.TEST.PS.'
6.6 Secure File Transfer Examples Using the z/OS Client

The client component of Tectia Server for IBM z/OS contains two file transfer applications, scp3 and sftp3.

- scp3 is a secure replacement for remote copy (rcp) and provides easy secure non-interactive file transfers.
- sftp3 is a secure replacement for FTP and provides a user interface for interactive file transfers and a batch mode for unattended file transfers.

The following examples (summarized in Table 6.3) can be used for interactive and unattended file transfers to and from a mainframe running Tectia Server for IBM z/OS 6.6. The same examples apply to file transfers against Windows and Unix servers. In these examples, the ssh_ftadv_config.example file transfer profiles are used. See Section 9.4.2 for more information.

Table 6.3. Examples of interactive and unattended secure file transfers using the scp3 and sftp3 clients.

<table>
<thead>
<tr>
<th>Interactive file transfers</th>
<th>Unattended file transfers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>scp3</strong></td>
<td></td>
</tr>
<tr>
<td>1. Remote Unix file into MVS data set</td>
<td>1. Remote file into MVS data set</td>
</tr>
<tr>
<td>2. MVS data set to remote Unix file</td>
<td>2. MVS data set to remote Windows file</td>
</tr>
<tr>
<td>3. Remote Windows text file into PDS member</td>
<td>3. MVS data set to remote Unix file using translate table</td>
</tr>
<tr>
<td>4. Remote Windows text file into fixed block PDS member</td>
<td>4. Remote file into pre-allocated MVS data set</td>
</tr>
<tr>
<td>5. z/OS binary file to another z/OS system</td>
<td>5. Remote Unix ASCII file into PDS member</td>
</tr>
</tbody>
</table>

| **sftp3**                   |                          |
| 1. Unix file to z/OS VSAM ESDS data set using filename-matched profiles | 1. Remote ASCII file into data set member |
| 2. File listing and several interactive file transfers between z/OS and Unix | 2. Run sftp3 using a batch command file |

6.6.1 Interactive File Transfers

Interactive file transfers can be used from Unix System Services shells, for example, OMVS, Telnet, or Secure Shell sessions can be used.
Note

In these examples, the ssh_ftadv_config.example file transfer profiles are used. See Section 9.4.2 for more information.

File Transfers Using the scp3 z/OS Client

The scp3 syntax is the following:

```
$ scp3 user@source:source_file user@destination:destination_file
```

Local paths can be specified without the user@system: prefix.

Note

In the following examples, data set names that contain single quotes and/or parentheses are placed in regular quotation marks (" "). This must be done to prevent the shell from trying to interpret the single quotes and parentheses. Alternatively, you can use backslashes (\) to escape the single quotes and parentheses, for example //\'USER1.WINPDS\(MEM1\)\".

Example 1: Fetch a Unix file into an MVS data set

A file transfer profile is not defined in the file transfer command, so the filename-matched profile is used if matched. In this case, the file name does not match any of the defined profiles, so the default profile is used (text format with code set conversion).

```
> scp3 user1@10.1.70.193:source_file //FILE1.PS
```

or

```
$ scp3 user1@10.1.70.193:source_file //\'USER1.FILE2.PS\'
```

Example 2: Put an MVS data set to a Unix file

A file transfer profile is not defined in the file transfer command, so the filename-matched profile is used if matched. The data set name has the ".Z" extension, so the correct profile is selected automatically (binary file transfer).

```
$ scp3 //\'USER1.PDS.Z\' user1@10.1.70.193:/tmp/binaries/file.Z
```

Example 3: Fetch a Windows text file into a partitioned data set member

In this case, a Windows profile is used in order to do the Windows line delimiter conversion correctly. The profile also defines code set conversion.

Using a file transfer advice string:

```
$ scp3 user1@10.1.70.100:textfile /ftadv:P=WIN://\'USER1.WINPDS\(MEM1\)\'
```

Or, alternatively, using a site parameter:
Example 4: Fetch a Windows text file into a fixed block partitioned data set member

A Windows profile is used for code set and line delimiter conversions, but additional parameters are required for defining the Fixed Block file format.

Using a file transfer advice string:

```
$ scpg3 user1@10.1.70.100:jcl-file /ftadv:P=WIN,O=FB,R=80///'USER1.WINPDS(JCL)'
```

Or, alternatively, using site parameters:

```
$ scpg3 --dst-site="P=WIN,O=FB,R=80" user1@10.1.70.100:jcl-file //''USER1.WINPDS(JCL)'''
```

Example 5: Copy a z/OS binary file to another z/OS system

To ensure that both parties handle the data set as binary, set the binary profile (P=BIN) or binary settings (X=BIN,F=STREAM) to both local and remote data sets. If you are not sure whether the profiles are enabled, use the binary settings (X=BIN,F=STREAM).

Using a file transfer advice string:

```
$ scpg3 /ftadv:P=BIN///LOCAL.BINARY \
    user@lpar2.example.com:/ftadv:X=BIN,F=STREAM///REMOTE.BINARY
```

Or, alternatively, using site parameters:

```
$ scpg3 --src-site="P=BIN" --dst-site="X=BIN,F=STREAM" //LOCAL.BINARY \
    user@lpar2.example.com://REMOTE.BINARY
```

File Transfers Using the sftpg3 z/OS Client

`sftpg3` has the `sput` and `sget` commands that can be used for mainframe file transfers.

Example 1: Fetch a Unix file to a z/OS VSAM ESDS data set using filename-matched profiles

An `sftpg3` connection is opened and a file is transferred from Unix to z/OS with the `sget` command.

The example assumes that you have set a filename-matched profile that matches `textfile.txt`.

```
$ sftpg3 user1@10.1.70.193
user1@10.1.70.193's password: ...

sftp> sget textfile.txt /ftadv:T=ESDS///FILE1.VSAM
textfile.txt | 49B | 49B/s | TOC: 00:00:01 | 100%
sftp> quit
```

Example 2: File listing and several interactive file transfers between z/OS and Unix

```
$ sftpg3 user1@10.1.70.193
user1@10.1.70.193's password: ...
sftp> ls
```

© 2007–2020 SSH Communications Security Corporation
mainframe_files/
source_file
textfile.txt

sftp> cd mainframe_files ❷
/home/user1/mainframe_files
sftp> ls ❸
/home/user1/mainframe_files:
binary.dat
jcl
sftp> aget binary.dat //USER1.BINARY.FILE' ❹
binary.dat | 4.6kB | 4.6kB/s | TOC: 00:00:01 | 100%
sftp> ascii ❺
sftp> aget jcl /ftadv:O=FB R=80///'USER1.PDS(MEM1)' ❻
jcl | 98B | 98B/s | TOC: 00:00:01 | 100%
sftp> sput //FILE1.PS /tmp/result.txt ❼
FILE1.PS | 49B | 49B/s | TOC: 00:00:01 | 100%
sftp> binary ❽
sftp> sput //BINARY.FILE binary_file.dat ❾
BINARY.FILE | 4.6kB | 4.6kB/s | TOC: 00:00:01 | 100%

1 List files in current working directory.
2 Change current working directory to mainframe_files.
3 List files in current working directory.
4 Get a file to an MVS data set.
5 Set transfer mode to ASCII.
6 Get an ASCII file into an MVS data set member, using a file transfer advice string to set recfm FB and lrecl 80.
7 Put an MVS data set to an ASCII file.
8 Set transfer mode to binary.
9 Put an MVS data set to a binary file.

6.6.2 Unattended File Transfers

Unattended file transfers of MVS data sets can be executed in JCL by BPXBATCH, BPXBATSL, or oshell. sftpg3 uses the same syntax for interactive and unattended file transfers. sftpg3 has a batch mode for non-interactive file transfers.

Note

User interaction is not possible when using unattended file transfers.

Users must be set up to use a non-interactive authentication method for unattended use, such as 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 Section 4.2 and Section 4.9.1.

The sample scripts shown in this section can also be found in <HLQ>.V669.SAMPLIB.

File Transfers Using the scpg3 z/OS Client

Example 1: Fetch a remote file into an MVS data set

This example (SCPGET from SAMPLIB) executes scpg3 and copies a remote file (file.bin) into a data set (//'USER.TEST.BINFILE'). If the data set does not exist, it is created with default values recfm VB and lrecl 1024.

The stdout and stderr message files are printed to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.

```plaintext
//SCPGET EXEC PGM=BPXBATSL,REGION=0M
//STDPARM DD *
PGM /opt/tectia/bin/scpg3
   user@remote:file.bin
    //"USER.TEST.BINFILE"
//STDENV DD DSN=<HLQ>.V669.PARMLIB(SSHENV),DISP=SHR
//STDOUT DD SYSOUT=* 
//STDERR DD SYSOUT=* 
//STDIN DD DUMMY 
//
```

Example 2: Put an MVS data set to a remote Windows file

In this example (SCPPUT1 from SAMPLIB), scpg3 is executed to copy a data set to a remote file (test.list), converting the code set from IBM-1047 to ISO8859-1 and records to CR-LF delimited lines.

The stdout and stderr message files are printed to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.

```plaintext
//SCPPUT1 EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT
//STDPARM DD *
PGM /opt/tectia/bin/scpg3
  /ftadv:C=ISO8859-1,D=IBM-1047,I=DOS,J=MVS//__HLQ.TEST.LIST
  user@remote:test.list
//STDENV DD DSN=<HLQ>.V669.PARMLIB(SSHENV),DISP=SHR
//STDOUT DD SYSOUT=* 
//STDERR DD SYSOUT=* 
//STDIN DD DUMMY 
//
```
Example 3: Put an MVS data set to a remote Unix file using translate table

In this example (SCCPUT from SAMPLIB), scp3 is executed to copy a data set assigned to DD LIST to a remote file (test.list), performing code set translation via the 'TCPIP.STANDARD.TCPXLBIN' translate table.

The stdout and stderr message files are printed to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.

//SCCPUT EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT
//STDPARM DD *
PGM /opt/tectia/bin/scp3
   /ftadv:E=STANDARD,A=___TCPIP.%T.TCPXLBIN,F=LINE///DD:LIST
       user@remote:test.list
//STDENV DD DSN=<HLQ>.V669.PARMLIB(SSHENV),DISP=SHR
//STDOUT DD SYSOUT=* 
//STDERR DD SYSOUT=* 
//STDIN DD DUMMY 
//LIST DD DSN=&SYSUID..TEST.LIST,DISP=SHR
//

Example 4: Fetch a remote file into a pre-allocated MVS data set

In this example (SCPGET2 from SAMPLIB), a remote file (testfile) is copied into a pre-allocated data set, assigned to DD TEST.

The stdout and stderr message files are printed to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.

//SCPGET2 EXEC PGM=BPXBATSL,REGION=0M
//STDPARM DD *
PGM /opt/tectia/bin/scp3
   user@remote:testfile
   //DD:TEST
//STDENV DD DSN=<HLQ>.V669.PARMLIB(SSHENV),DISP=SHR
//STDOUT DD SYSOUT=* 
//STDERR DD SYSOUT=* 
//STDIN DD DUMMY 
//TEST DD DSN=&SYSUID..TEST,DISP=(NEW,CATLG),
    VOL=SER=ZZSYS1,SPACE=(TRK,(2,2)),
    DCB=(RECFM=VB,LRECL=1024,BLKSIZE=27998)
//

Example 5: Fetch a remote Unix ASCII file into a data set member

In this example (SCPGET3 from SAMPLIB), a remote Unix ASCII file (jcl.txt) is copied to mainframe into a PDS member using lrecl 80 and recfm FB, creating the PDS if it does not exist.

The stdout and stderr message files are printed to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.
Example 6: Fetch a remote Windows ASCII file into a data set member

In this example (SCPGET3W from SAMPLIB), a remote ASCII CRLF (Windows/DOS) file (jcl.txt) is copied to mainframe into a PDS member using lrecl 80 and recfm FB, creating the PDS if it does not exist.

The stdout and stderr message files are printed to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.

File Transfers Using the sftpg3 z/OS Client

The sftpg3 file transfer application can be run in batch mode for non-interactive file transfers.

Example 1: Fetch a remote ASCII file into a data set member

In this example (SFTPBAT from SAMPLIB), sftpg3 is run in batch mode to copy a remote ASCII file (jcl.txt) into a PDS member using lrecl 80 and recfm FB. If the PDS does not exist, it will be created.

The stdout and stderr message files are printed to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.
Example 2: Run sftpg3 using a batch command file

In this example (SFTP from SAMPLIB), sftpg3 is run in batch mode (with the -B option). Since it is not possible to enter a password, we use public-key authentication with no passphrase for the private key.

sftpg3 returns the highest error code as the return code from the run. BPXBATSL multiplies this by 256 and returns the product as the step's condition code.

Using the ls command may cause non-zero error codes, for example if a referenced data set is in use in another address space. Run listings in separate steps if it is important to get the proper condition codes from file transfers.

The stdout and stderr message files are printed combined to SYSOUT. Required environment variables are supplied in SSHENV via STDENV DD. Modify the DD statement according to your requirements.

1. Change current working directory to //USER.TEST.REPORTS
2. Fetch RPT1 to //TEST.RPT1
3. List the contents of //TEST.RPT1 in long format.
Chapter 7 Secure Shell Tunneling

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

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

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

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

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

7.1 Local Tunnels

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

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

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

where:

- `[protocol/]` specifies which protocol is to be used in the tunneled connection, it can be `ftp` or `tcp` (optional argument). The default is `tcp`.
• \texttt{[listen-address:] } defines which interface on the local client will be listened to (optional argument). By default all interfaces are listened.

• \texttt{listen-port} is the number of the port on the local client, and connections coming to this port will be tunneled to the server.

• \texttt{dst-host:dst-port} define the destination host address and the port to which the connection is tunneled from the server.

• \texttt{sshserver} is the IP address or the host name of the Secure Shell server.

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

\begin{itemize}
  \item \textbf{Note}
\end{itemize}

\begin{itemize}
  \item If \texttt{dst-host} is specified as a domain name rather than IP address, the name will be resolved according to the address family settings of \texttt{sshserver}. For example, if the domain name is resolved to an AAAA DNS record (IPv6) and the address family setting of the server is \texttt{inet} (IPv4), the tunnel will not work.
\end{itemize}

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

\begin{itemize}
  \item \textbf{Note}
\end{itemize}

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

\begin{itemize}
  \item \texttt{Figure 7.1} shows the different hosts and ports involved in local tunneling (port forwarding).
\end{itemize}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{local_tunneling_terminology.png}
\caption{Local tunneling terminology}
\end{figure}
For example, when you issue the following `sshg3` command on the command line, all traffic coming to port 1234 on the client host will be forwarded to port 23 on the server.

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

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

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

### 7.1.1 Non-Transparent TCP Tunneling

When non-transparent TCP tunneling is used, the application to be tunneled is set to connect to the local listener port instead of connecting to the server directly. Tectia client tools for z/OS forwards the connection securely to the remote server.

If you have three hosts, for example, `sshclient`, `sshserver`, and `imapserver`, and you forward the traffic coming to the `sshclient`'s port 143 to the `imapserver`'s port 143, only the connection between the `sshclient` and `sshserver` will be secured. The command you use would be similar to the following one:

```
sshclient$ sshg3 -L 143:imapserver:143 username@sshserver
```

Figure 7.3 shows an example where the Secure Shell server resides in the DMZ network. Connection is encrypted from the Secure Shell client to the Secure Shell server and continues unencrypted in the corporate network to the IMAP server.
Tunnels can also be defined for connection profiles in the Connection Broker configuration file. The defined tunnels are opened automatically when a connection with the profile is made. The following is an example from a `ssh-broker-config.xml` file:

```xml
<profile id="id1" host="sshserv.example.com">
  ...
  <tunnels>
    <local-tunnel type="tcp"
      listen-port="143"
      dst-host="imap.example.com"
      dst-port="143"
      allow-relay="no" />
    ...
  </tunnels>
</profile>
```

By default, local tunnels originating only from the client host itself are allowed. To allow also other machines to connect to the tunnel listener port, set the `allow-relay` to `yes`.

**Automatic Tunnels**

Automatic tunnels are one way of creating non-transparent local tunnels for application connections.

Automatic tunnels always use a connection profile in the tunnel establishing. You can create listeners for local tunnels that will be activated automatically when the Connection Broker starts up. The actual tunnel will be formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.

In the Connection Broker configuration file, make the following kind of settings:

```xml
<static-tunnels>
  <tunnel type="tcp"
    listen-port="9874"
    dst-host="st.example.com"
    dst-port="9111"
    allow-relay="no"
    allow-relay="no"
</tunnels>
```
Examples of Local Tunneling

When sshg3 is used to create secure tunnels using local port forwarding, the TCP applications to be tunneled are configured to connect to a localhost port instead of the application server port.

Example application, clientapp1, by default connects to a Unix server unix.example.com using TCP port 2345.

$ clientapp1 --username user1 --server unix.example.com --port 2345

For securing this TCP application using Secure Shell, use the following commands:

$ sshg3 -L 2345:localhost:2345 user1@unix.example.com -S -f &
$ clientapp1 --username user1 --server localhost --port 2345

The above sshg3 command connects to remote Secure Shell server unix.example.com, creates a local listener on port 2345, instructs the remote Secure Shell server to forward the incoming traffic to localhost:2345, and goes to background in single-shot-mode.

7.1.2 Non-Transparent FTP Tunneling

Non-transparent FTP tunneling is an extension to the generic tunneling mechanism. Unlike generic tunneling (port forwarding) mechanism, non-transparent FTP tunneling secures the transferred files, in addition to the FTP control channel. The FTP tunneling code monitors the tunneled FTP control channels and dynamically creates new tunnels for the data channels as they are requested.

When non-transparent FTP tunneling is used, tunnels are created from local client ports to remote servers. The FTP client is configured to connect to Tectia client tools for z/OS which will forward the connection to the endpoint where a Secure Shell server is running.

The typical use case is that Tectia client tools for z/OS is located on the same host as the FTP client; and the FTP server is on the same host as the Secure Shell server. However, other configurations are also supported, but it is worth noticing that the connection is encrypted only between Tectia client tools for z/OS and the Secure Shell server.

Non-transparent FTP tunneling can be requested on the command line, or enabled and defined in the Connection Broker configuration. The configured non-transparent FTP tunneling uses connection profiles, that are defined on Tectia client tools for z/OS.

On command-line, FTP tunneling can be used for both local and remote tunnels. Non-transparent FTP-tunneling is started by entering a sshg3 command with the following syntax:

sshclient$ sshg3 -L ftp/1234:localhost:21 username@sshserver

For information on the sshg3 command, see the sshg3(1) man page.
FTP tunnels can also be defined for connection profiles in the Connection Broker configuration file. The following is an example from the Connection Broker configuration file ssh-broker-config.xml:

```xml
<profiles>
  <profile id="id1" host="sshserver.example.com"
    ...
  <tunnels>
    <local-tunnel type="FTP"
      listen-port="1234"
      dst-host="127.0.0.1"
      dst-port="21"
      allow-relay="NO" />
    ...
  </tunnels>
  </profile>
</profiles>
```

An FTP connection can then be made with the following (example) commands:

```
sshclient$ ftp
tftp$ open localhost 1234
```

The FTP connection to port 1234 on client is now tunneled to port 21 on the Secure Shell server.

As an alternative to FTP tunneling, you can use the **sftpg3** or **scpg3** clients for secure file transfers. These clients can be used on command line or in scripts and they require less configuration than FTP tunneling, since Tectia Server already has **sft-server-g3** as a subsystem, and **sftpg3** and **scpg3** clients are included with Tectia client tools for z/OS. Managing remote user restrictions on the server machine will be easier, since you do not have to do it also for FTP.

To understand exactly how FTP tunneling is done, two different cases need to be examined: the active mode and the passive mode of the FTP protocol.

### Tunneling FTP in Passive Mode

In passive mode, the FTP client sends the command *PASV* to the server, which reacts by opening a listener port for the data channel and sending the IP address and port number of the listener as a reply to the client. The reply is of the form

```
227 Entering Passive Mode (a1,a2,a3,a4,p1,p2)
```

where `a1.a2.a3.a4` is the IP address and `p1*256+p2` is the port number. For example, the reply for IP address `10.1.60.99` and port `1548` is: `227 Entering Passive Mode (10,1,60,99,6,12)`.

When the Connection Broker notices the reply to the *PASV* command, it will create a local port forwarding to the destination mentioned in the reply. After this the Connection Broker will rewrite the IP address and port in the reply to point to the listener of the newly created local port forwarding (which exists always in a localhost address, `127.0.0.1`) and pass the reply to the FTP client. The FTP client will open a data channel based on the reply, effectively tunneling the data through the Secure Shell connection, to the listener the FTP server has
opened. The net effect is that the data channel is secured all the way except from the Secure Shell server to
the FTP server if they are on different machines. This sequence of events takes place automatically for every
data channel.

Since the tunnel is opened to a localhost address, the FTP client must run on the same machine as Tectia client
tools for z/OS if passive mode is used.

**Tunneling FTP in Active Mode**

In active mode, the FTP client creates a listener on a local port, for a data channel from the FTP server to the
FTP client, and requests the channel by sending the IP address and the port number to the FTP server in a
command of the following form:

```
PORT a1,a2,a3,a4,p1,p2
```

where `a1.a2.a3.a4` is the IP address and `p1*256+p2` is the port number. The Connection Broker intercepts
this command and creates a remote port forwarding from the localhost address of the Secure Shell server to
the address and port specified in the `PORT` command.

After creating the tunnel, the Connection Broker rewrites the address and port in the `PORT` command to point
to the newly opened remote forwarding on the Secure Shell server and sends it to the FTP server. Now the
FTP server will open a data channel to the address and port in the `PORT` command, effectively forwarding the
data through the Secure Shell connection. The Connection Broker passes the incoming data to the original
listener created by the FTP client. The net effect is that the data channel is secure the whole way except from
Tectia client tools for z/OS to the FTP client. This sequence of events takes place automatically for every data
channel.

For FTP tunneling in active mode to work, the FTP server must be run on the same host as the Secure Shell
server, and the FTP client and Tectia Client must reside on the same host.

**Note**

Tunneling FTP in active mode is not guaranteed to work in all setups. If possible, use the passive
mode when tunneling FTP connections.

**7.1.3 SOCKS Tunneling**

SOCKS tunneling is a mechanism available for tunneling applications that support the SOCKS4 or SOCKS5
client protocol.

Instead of configuring tunneling (a.k.a port forwarding) from specific ports on the local host to specific ports
on the remote server, you can specify a SOCKS server which can be used by the user's applications. Each
application is configured in the regular way except that it is configured to use a SOCKS server on a localhost
port. The Secure Shell client application, Tectia client tools for z/OS, opens a port in the localhost and mimics
a SOCKS4 and SOCKS5 server for any SOCKS client applications.
When the applications connect to services such as IMAP4, POP3, SMTP, and HTTP, they provide the necessary information to the SOCKS server, which is actually Tectia client tools for z/OS mimicking a SOCKS server. Tectia client tools for z/OS will use this information in creating a tunnel to the Secure Shell server and relaying the traffic back and forth securely.

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

```
client$ sshg3 -L socks/[listen-address:]listen-port username@sshserver
```

where:

- `[listen-address:]` defines which interface on the client will be listened to (optional argument)
- `listen-port` is the number of the port on the client
- `sshserver` is the IP address or the host name of the Secure Shell server.

For example, the following command will set up a local tunnel from port 1234 on the client to `sshserver`. The applications are set to use a SOCKS server at port 1234 on the client. From the server, the connections are forwarded unsecured to the destination hosts requested by the applications.

```
sshclient$ sshg3 -L socks/1234 username@sshserver
```

SOCKS tunnels can also be defined for connection profiles in the Connection Broker configuration file. The following is an example from a `ssh-broker-config.xml` file:

```xml
<profile id="id1" host="sshserver.example.com">
  ...
  <tunnels>
    <local-tunnel type="socks"
      listen-port="1234"
      allow-relay="no" />
  ...
  </tunnels>
</profile>
```

### 7.2 Remote Tunnels

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

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

```
client$ sshg3 -R [protocol/] [listen-address:]listen-port:dst-host:dst-port \ username@sshserver
```

where:

- `[protocol/]` specifies which protocol is to be used in the tunneled connection, it can be `ftp` or `tcp` (optional argument). The default is `tcp`. 
• **[listen-address:]** defines which interface on the remote server will be listened to (optional argument). By default all interfaces are listened.

• **listen-port** is the number of the port on the remote server, and connections coming to this port will be tunneled to the client.

• **dst-host:dst-port** define the destination host address and the port to which the connection is tunneled from the client.

• **sshserver** is the IP address or the host name of the Secure Shell server.

The IP addresses and host names of the destination host and the sshserver can be defined using regular expressions that follow the egrep syntax. No wildcards are supported.

**Note**

If **dst-host** is specified as a domain name rather than IP address, the name will be resolved according to the address family settings of **client**. For example, if the domain name is resolved to an AAAA DNS record (IPv6) and the address family setting of the client is **inet** (IPv4), the tunnel will not work.

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

**Note**

Every user with access to the remote server host will be able to use remote tunnel.

**Figure 7.4** shows the different hosts and ports involved in remote port forwarding.

**Figure 7.4. Remote tunneling terminology**
For example, if you issue the following command, all traffic which comes to port 1234 on the server will be tunneled to port 23 on the client. See Figure 7.5.

```
sshclient$ sshg3 -R 1234:localhost:23 username@sshserver
```

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

![Figure 7.5. Remote tunnel](image)

Tunnels can also be defined for connection profiles in the Connection Broker configuration file. The defined tunnels are opened automatically when a connection with the profile is made.

The following is an example from a `ssh-broker-config.xml` file:

```xml
<profile id="id1" host="sshserver.example.com">
  ...
  <tunnels>
    <remote-tunnel type="tcp"
      listen-port="1234"
      dst-host="localhost"
      dst-port="23" />
  ...
  </tunnels>
</profile>
```

### 7.3 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.

Tectia client tools for z/OS provides authentication agent functionality and the Connection Broker can also serve OpenSSH clients as an authentication agent. Tectia Server supports agent forwarding on Unix platforms. Thus, the start and end points of the agent forwarding chain can be Windows or Unix hosts, but all hosts in the middle of the forwarding chain must be Unix hosts and must have both the Secure Shell client and server components installed.
Figure 7.6. Agent forwarding

In the factory settings, agent forwarding is enabled (on).

Agent forwarding can be enabled or disabled on the client side both in the default configuration settings and separately for each connection profile.

In the `ssh-broker-config.xml` file, agent forwarding can be disabled by setting the following line either under `default-settings` or under a connection profile:

```
<forwards>
  <forward type="agent" state="off"/>
</forwards>
```
Chapter 8 Troubleshooting Tectia

You can run Tectia Client in debug mode to gather information that is useful when troubleshooting any connection or authentication problems.

Do not leave the client running in debug mode unnecessarily. Debugging slows down the performance of the client.

See Section 1.2 for information on accessing the Tectia online support resources and contacting Tectia Technical Support.

8.1 Starting Connection Broker in Debug Mode

The Connection Broker is an internal component included in Tectia Client and Tectia ConnectSecure. The Connection Broker handles all cryptographic operations and authentication-related tasks for Tectia Client, ConnectSecure, and the command-line tools `sshg3`, `scpg3`, `sftpg3`.

To start the Connection Broker in debug mode, follow these instructions:

1. Open a shell (on Unix) or command prompt window (on Windows).

2. Stop the Connection Broker, if it is currently running. Enter the following command to exit the Connection Broker. This will close all currently open connections of the current user:

   ```
   $ ssh-broker-g3 --exit
   ```

3. Start the Connection Broker in debug mode by running the following command:

   ```
   $ ssh-broker-g3 -D<filter> -l <logfile>
   ```

   In the command:
   - `logfile` specifies the file to which the debug output will be directed
   - `filter` is an expression that takes the following syntax: "module-level,module-level,..."
• **module** is an optional expression that can be used to restrict the debug output to only a particular module or to allow the use of varying debug levels for different modules.

• **level** is an integer from 0 (no debug info) to 99 that specifies the desired amount of debug information.

Note that levels 1-9 are the recommended ones. The higher the number, the more detailed the troubleshooting output will be, and the more the debugging will affect performance.

The following example command starts the Connection Broker with global debug level 4 and outputs the debug information to a log file named `broker.log`:

```bash
$ ssh-broker-g3 -D4 -l broker.log
```

The following example command starts the Connection Broker with debug level 5 for modules starting with "SecShAuth" and level 2 for everything else:

```bash
$ ssh-broker-g3 -D"SecShAuth*=5,2" -l broker.log
```

4. Connect to a server using one of the clients:

```bash
$ sshg3 user@host
```

5. View the debug information for the connection in the `broker.log` file.

On Unix, you can display the debug output also by using the command line tools with argument `-D`. For example, the following command will display the debug output with a debug level 5 for modules starting with `SecShAuth` and level 2 for modules starting with `Sft`:

```bash
$ sftpg3 -D"SecShAuth*=5,Sft*=2" user@host
```

### 8.2 Debugging File Transfer

The return codes and error messages should be sufficient to explain most errors. Check that the code and message are correct by looking at the results:

• Was the file transferred OK but the error code wrong?

• Was the error code 0 but the file corrupted?

To diagnose file transfer problems, take a trace. If possible, also take a trace of a similar but successful run and compare the two traces. To get traces that can be compared with a `diff` command, use a trace format without "%Dx" or "%p" specifiers:

```bash
SSH_DEBUG_FMT=" %m/%s:%n:%f %M"
```

To trace the generic file transfer level, use the debug string `sftp*=14`. 
To trace the z/OS low level, use the debug string '*mvs*=7'.

To trace both, use 'sftp*=14,*mvs*=7'.

To get the trace for the client end, put the debug string on the command line, for example:

```
$ sftpg3 -D "sftp*=14" user1@www.example.com
```

To get the trace for the client end when using `sftpg3`, use the `debug` command:

```
sftp> pwd
sftp> lpwd
sftp> debug sftp*=14,*mvs*=7
sftp> sput tmp.txt //TMP.TXT
sftp> debug 1
```

To get the trace for the server end when using `sftp/scp`, create, on the remote machine, the file $HOME/.ssh2/environment, and put in a line like this:

```
SSH_SFTP_DEBUG=sftp*=14
```

To make the debug at the server end go into a HFS file at the server, add this line in the $HOME/.ssh/environment file:

```
SSH_SFTP_DEBUG_FILE=/tmp/my-debug.txt
```

---

### 8.3 Solving Problem Situations

This section gives workaround instructions on common error situations that may occur when using Tectia client tools for z/OS or when using other Secure Shell clients to connect to Tectia Server for IBM z/OS.

#### Interactive Login from Unix to z/OS

When interactively logging in from a Unix (usually HP-UX) client host to an Tectia Server for IBM z/OS host, the terminal is garbled and the shell does not accept input properly.

This happens because user terminals on Unix (especially HP-UX) often have the `istrip` (strip-8th-bit) flag on. This setting gets inherited by the pseudo terminal on z/OS which causes an interactive /bin/sh session malfunction.

To avoid the anomaly, do one of the following steps:

- Give the command "`stty -istrip`" on the Unix host before connecting the z/OS host with Secure Shell.

  **OR**

- Call "`stty -istrip`" on the z/OS host in the shell init file ($HOME/.profile or equivalent).

To avoid surprises, check that the shell has a `tty` before calling `stty`:
if [ tty >/dev/null 2>&1 ]; then
    stty -istrip
fi
Chapter 9 Using Off-Platform Clients to Access z/OS Hosts Running Tectia Server for IBM z/OS

This chapter contains information on using Secure Shell clients other than Tectia client tools for z/OS to connect to Tectia Server for IBM z/OS:

- Public-key authentication from other hosts to Tectia Server for IBM z/OS
- Handling MVS data sets and HFS file system access from other hosts to Tectia Server for IBM z/OS
- File transfer staging from other hosts to Tectia Server for IBM z/OS
- Listing data sets with other SFTP clients
- File transfer advice strings and file transfer profiles
- Examples on file transfer using Windows and Unix clients

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

The default OpenSSH configuration on z/OS does not produce SMF records. SMF recording must be configured separately for OpenSSH if the OpenSSH SCP events are intended to be captured.

9.1 Using Public-Key Authentication from Other Hosts to z/OS

In the following instructions:

- `Server_zos` is the remote host running Tectia Server for IBM z/OS that you are trying to connect to.
• **ServerUser** is the user name on **Server_zos** that you are logging in as.

• **Client** is the host running the Secure Shell client.

• **ClientUser** is the user name on **Client** that should be allowed to log in to **Server_zos** as **ServerUser**.

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

### 9.1.1 From Tectia Client on Windows to Tectia Server on z/OS

These instructions apply to Tectia Client and Tectia ConnectSecure on Windows. For more information, see *Tectia Client User Manual* or *Tectia ConnectSecure Administrator Manual*.

**Using Graphical User Interface**

On Windows and Linux, you can use the Tectia Public-Key Authentication Wizard to generate a key pair.

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

   The wizard will generate two key files, your private key and your public key. The private key file has no file extension, and the public key has the same base file name as the private key, but with `.pub` as the file extension. The key files will be stored on your local computer, in the user profile directory.

2. Public keys can be uploaded automatically to servers that have the SFTP subsystem enabled. The automatic upload can be done on the **Keys and Certificates** page of the **Tectia Connections Configuration GUI**.

   Select your key pair from the list and click **Upload**. The **Upload Public Key** dialog box opens.

3. Define the remote host where you want to upload the key.
Figure 9.1. Uploading the public key

Click **Upload** to start the upload.

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

5. Make sure that public-key authentication is allowed in the Connection Broker configuration on **Client**, in the default settings and in the relevant connection profile (it is allowed by default).

**Using Command-Line Tools**

The key pair can also be generated and transferred to the z/OS server by using command-line tools.

1. Create a key pair using **ssh-keygen-g3**. For non-interactive use, the key can be generated without a passphrase with the **-p** option. The **-t** option can be used to specify the key type (the default is RSA).

   ```
   C:\>ssh-keygen-g3 -P win_key
   Generating 2048-bit rsa key pair
   9 oOo.oOo.oOo
   Key generated.
   2048-bit rsa, ClientUser@TECTIA_WIN, Wed Feb 2 2016 12:09:46 +0200
   Private key saved to C:\Users\ClientUser\AppData\Roaming\SSH\UserKeys\win_key
   Public key saved to C:\Users\ClientUser\AppData\Roaming\SSH\UserKeys\win_key.pub
   ```

2. Create a remote **.ssh2** directory on **Server_zos** (if it does not exist already):
3. Transfer the public key to ```Server_zos``` with conversion options:

```bash
C:\>scp -a "C:\Users\ClientUser\AppData\Roaming\SSH\UserKeys\win_key.pub"
ServerUser@Server_zos:˜/.ssh2/
```

4. Create the remote authorization file on ```Server_zos```:

```bash
C:\>ssh ServerUser@Server_zos "echo Key win_key.pub >> .ssh2/authorization"
```

5. Make sure that public-key authentication is allowed in the Connection Broker configuration on ```Client``` in the default settings and in the relevant connection profile (it is allowed by default).

### 9.1.2 From Tectia Client on Unix to Tectia Server on z/OS

These instructions apply to Tectia Client and Tectia ConnectSecure on Unix. For more information, see [Tectia Client User Manual](#) or [Tectia ConnectSecure Administrator Manual](#).

To enable public-key authentication from Tectia Client on Unix to Tectia Server on z/OS:

1. Create a key pair using ```ssh-keygen-g3```.

```
$ ssh-keygen-g3 -t rsa -P $HOME/.ssh2/unix_key
Generating 2048-bit rsa key pair
   .o.O.o.O.o.O.o.O
Key generated.
2048-bit rsa, ClientUser@tectia_unix, Tue Nov 11 2014 10:43:23 +0200
Private key saved to /home/ClientUser/.ssh2/unix_key
Public key saved to /home/ClientUser/.ssh2/unix_key.pub
```

2. Create a remote .ssh2 directory on ```Server_zos``` (if it does not exist already):

```
$ ssh ServerUser@Server_zos mkdir .ssh2
```

3. Copy your public key to the remote ```Server_zos```:

```
$ scp -a unix_key.pub \
ServerUser@Server_zos:˜/.ssh2/unix_key.pub
```

4. Create an authorization file on the remote ```Server_zos```.

```
$ ssh ServerUser@Server_zos "echo Key unix_key.pub >> .ssh2/authorization"
```

5. Make sure that public-key authentication is allowed in the Connection Broker configuration on ```Client``` in the default settings and in the relevant connection profile (it is allowed by default).
9.1.3 From OpenSSH Client on Unix to Tectia Server on z/OS

In addition to the standard IETF SecSh keys used by Tectia, Tectia Server for IBM z/OS accepts OpenSSH public keys for user authentication. For more information on OpenSSH configuration, see OpenSSH documentation.

To enable public-key authentication from OpenSSH client on Unix to Tectia Server on z/OS:

1. Create a key pair using `ssh-keygen`, for example:

   ```
   $ ssh-keygen -t rsa
   Generating public/private rsa key pair.
   Enter file in which to save the key (/home/ClientUser/.ssh/id_rsa):
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in /home/ClientUser/.ssh/id_rsa.
   Your public key has been saved in /home/ClientUser/.ssh/id_rsa.pub.
   The key fingerprint is:
   The key's randomart image is:
   +--[ RSA 2048]----+
   |                 |
   |                 |
   |      .          |
   |   + . . .       |
   |  o E o S o      |
   | .   . = = .     |
   |  . .   = B      |
   | .o    .=+       |
   +-----------------+
   ```

   To create the key without a passphrase, hit enter when prompted to enter the passphrase.

   When the key is created with default file name (id_rsa), it is automatically used in public-key authentication attempts.

2. Create a `.ssh2` directory on `Server_zos` (if it does not exist already):

   ```
   $ ssh ServerUser@Server_zos mkdir .ssh2
   ```

3. Copy your public key to the remote `Server_zos` using `sftp`:

   ```
   $ sftp ServerUser@Server_zos
   sftp> put id_rsa.pub /ftadv:C=ISO8859-1,D=IBM-1047,X=TEXT/.ssh2/id_rsa.pub
   ```

4. Create an authorization file on the remote `Server_zos`.

   ```
   $ ssh ServerUser@Server_zos "echo Key id_rsa.pub >> .ssh2/authorization"
   ```
5. Make sure that public-key authentication is allowed in the OpenSSH client configuration on client (it is allowed by default).

9.2 Setting up Terminal Data Conversion

A remote account on a z/OS machine can have a locale and a code page that differs from the default 'C' locale and IBM-1047 code page. Tectia Server for IBM z/OS can convert the terminal data stream to match the locale.

The code page can be specified in the server configuration or with the `chcp` command (see the z/OS UNIX System Services Command Reference). The `chcp` command is typically used in a `.profile` or `.tshrc` script, see the IBM book z/OS USS User's Guide), but it can also be used during an interactive session.

For remote command sessions, the code page can be specified in the configuration of Tectia Server for IBM z/OS. It is possible to define user-id-specific and client-host-specific subconfigurations, see the Tectia Server for IBM z/OS Administrator Manual.

Most SSH clients allow the user to specify how line delimiters are to be handled. Tectia Server for IBM z/OS can also be configured for EOL conventions.

The code page of the data on the line is specified as an argument in the `chcp` command and in the server configuration. The default is Latin 1, (ISO8859-1). If the SSH client uses some other code page, the code page should be specified in the `chcp` command or in the configuration.

9.3 Handling MVS Data Sets and HFS File System Access

This section describes how access to MVS data sets and HFS files are handled when other clients than Tectia client tools for z/OS are used to connect to Tectia Server for IBM z/OS.

9.3.1 Data Set and HFS File System Access

Old SFTP clients regard `/` (slash) as a directory separator, which requires the SFTP server to use some "tricks" to fool the clients. The SFTP client generally does not have any inherent knowledge of MVS data sets and accesses all files as if they were part of a hierarchical file system.

If an SFTP client tries to open the file `//FILE.NAME`, the MVS data set is actually opened. Some SFTP clients remove all consecutive slashes so that the file to be accessed becomes `/FILE.NAME`. The SFTP server will interpret this as the HFS file `FILE.NAME`, located at the HFS root. To avoid this client behavior, the SFTP server has other ways of naming MVS data sets. If a client tries to access the file `/_FILE.NAME`, `/FTADV:X=BIN/_FILE.NAME` or `/FTADV:X=BIN/_FILE.NAME`, the SFTP server interprets these as the data set `//FILE.NAME`. All these combinations are needed as it may be necessary to make the SFTP clients pass the correct file or data set name to the server.
In z/OS, if a data set name is not enclosed in single quotes, the user prefix is added in front of the data set name. For example, if user USER1 has a data set DATASET.NAME1, the user can access it using the data set name //DATASET.NAME1. It is also possible to use an absolute prefixed name //USER1.DATASET.NAME1.

Because single quotes are special characters in most environments, the SFTP server introduces a new convention to use triple slashes to access absolute prefixed data set names. The name becomes //USER1.DATASET.NAME1. All combinations of slashes and underscores can be used. For example, USER1 can access his/her data set using one of the following names:

```
//DATASET.NAME1
/_DATASET.NAME1
/FTADV:X=BIN/_/DATASET.NAME1
/FTADV:X=BIN/__DATASET.NAME1
```

or

```
//USER1.DATASET.NAME1'
/_USER1.DATASET.NAME1'
/FTADV:X=BIN/_/USER1.DATASET.NAME1'
/FTADV:X=BIN/__USER1.DATASET.NAME1'
```

or

```
///USER1.DATASET.NAME1
//USER1.DATASET.NAME1
/_USER1.DATASET.NAME1
/_USER1.DATASET.NAME1
/FTADV:X=BIN/_/_USER1.DATASET.NAME1
/FTADV:X=BIN/_//USER1.DATASET.NAME1
/FTADV:X=BIN/__USER1.DATASET.NAME1
/FTADV:X=BIN/__USER1.DATASET.NAME1
```

The choice of name type depends on the client. If consecutive slashes are not sent to the server, "/_" (or "/_" for a prefixed absolute data set) is probably the best choice. The SFTP client interprets the file path to be absolute and does not add any directories in front of the name. If a file transfer advice string (see Section 9.4.1) is added in front of the file name, "__" (or "___" for a prefixed absolute data set) is a safe choice. The data set name becomes /FTADV:X=BIN/__DATASET.NAME1. The client interprets it as an absolute path to the file and does not change it. The SFTP server on the other hand recognizes the advice string /FTADV:X=BIN/ and interprets __DATASET.NAME1 to be the real name of the data set //DATASET.NAME1.

z/OS has also library data sets, whose members are accessed using the data set name //DATASET.NAME1(MEMBER1).

Again, parentheses may be special characters in some environments. The SFTP server introduces a new convention for accessing library members using a single slash as a member separator. The name becomes //DATASET.NAME1/MEMBER1.
Accessing HFS Root

Because of the new conventions for data set names there are some exceptions in the HFS file names. HFS file or directory name cannot start with an underscore if the file or directory is located in the root directory. Also, if an advice string is used with absolute HFS names, the HFS root must be written either as / or _.

Case-Sensitivity of HFS and MVS Names

HFS file names are case-sensitive. For example, /tmp/MYFILE and /tmp/myfile result in two different files.

MVS data set names are case-insensitive. For example //USER1.DATASET.NAME1' and //user1.dataset.name1' are handled the same way.

9.3.2 Accessing Generation Data Groups (GDG)

Tectia Server for IBM z/OS supports generation data groups defined in ICF. Reading GDG ALLs is not supported. Tectia Server for IBM z/OS will not create GDG bases or model data sets.

Note

Tectia file transfers are atomic. Running "sget gdg.base(0) file1" two times in succession might retrieve different files. A loop of "sput file1 gdg.base(+1)" commands might fill the GDG with identical files and roll off all the previous generations.

In the examples below, sshd2 means server functions on z/OS that can be used by any remote client.

Navigating

sshd2 allows you to navigate to a GDG base and to a prefix of a base. The name of the GDG base is returned to the client with a period at the end, the same way as for prefixes.

Listing

sshd2 shows GDG bases with the type GDG. Generation data sets (GDS) are shown with their absolute data set names.

GDSs are normal data sets. The long name format (ls -l) will show all details.

Listing a base with -1 will show full details of the GDSs. The listing may contain data sets that are not in the GDG index but do have data set names that have the GDG name as a prefix.

It is possible but not recommended to use data set names which have the GDG base name as a prefix but are not GDS names. For example:

```bash
sftp> cd //USER1.GENGRP
MVS prefix `USER1.GENGRP.' is the current directory.
The working directory `USER1.GENGRP.' is a generation data group.
```
You cannot navigate to a prefix that ends in a GnnnnVnn qualifier. Thus you cannot do "cd G0006V00" or "ls //’USER1.GENGRP.G0006V00’" in the example above.

If the GDG has the NOSCRATCH option, GDSs are retained when they are rolled off.

sshd2 shows data sets based on the prefix – it does not show which data sets are in the GDG and which are not.

**Access by Relative DSN**

sshd2 gives full access with relative generation numbers for reading and writing.

The following formats can be used for the relative GDS name abc.xyz{relno} (relno must be 0, +1, or -nn):

- abc.xyz{relno}
- abc.xyz./relno
- abc.xyz/relno
- abc.xyz.relno

**Access by Absolute DSN**

With absolute GDS names you can do all the things possible with other data sets.

Writing a data set with a last qualifier with the GnnnnVnn format requires that there exists a suitable GDG base. If the generation exists it is overwritten. If it does not, the new file is inserted in its place in the GDG and older GDGs are rolled off, if necessary.

**9.4 Alternate Methods for 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 CCS.

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. 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.

Using the `site` command is described in Section 6.4.1. Using environment variables is described in Section 6.4.2 and Section 9.4.3. For information on using third-party clients, see Section 9.4.1 and Section 9.4.2.

### 9.4.1 File Transfer Advice String (FTADV)

When using a third-party Secure Shell client for transferring files to and from Tectia Server for IBM z/OS, it is necessary to relay information about the transfer format, code sets and file characteristics. Since the SFTP server decodes the name for the data set, the file name in the request can be used to convey this information.

Instructions for using file transfer advice string, as well as a complete description of the available advice string names, are provided in Section 6.4.1.

#### Example Advice Strings

The following examples demonstrate file transfer advice string usage in various situations.

**Example 1**

Below is an example of a file name that requests the transfer of a PDS member with the line transfer format and code set conversion from EBCDIC to an ASCII code set.

```
/ftadv:D=IBM-1047,C=ISO8859-1,F=line/__personal.cntl/idlist
```

**Example 2**

The following example requests the transfer of a data set with the line transfer format and code set conversion using the translate table `USR1.SFTP.TCPXLBIN`, if it exists, or `TCP1P.SFTP.TCPXLBIN`. Different combinations of underscore and slash ("_", "/", "/_", or "/") in front of the file name indicate that the file is an MVS data set.

```
/ftadv:F=line,E=SFTP,A=___USR1.%T.TCPXLBIN+___TCP1P.%T.TCPXLBIN/__DATA1.FILE1
```

**Example 3**

The example below names an HFS file to be transferred without changes. Transfer mode is set to binary (X=bin) to avoid conversion and to override any defaults set in the matching file transfer profiles or environment files.

```
/ftadv:X=bin,T=HFS,F=stream/profcopy
```

**Example 4**

The next example uses the named file transfer profile `myprofile`. The advice string also sets the data set code set to `ISO8859-15`. This value overwrites the value specified in the profile.
Note

Advice strings can also be used with directory names in file transfer GUIs. With advice strings, file transfer can be controlled in the GUI. For example in the Windows GUI, if you use the directory 
/ftadv:P=WIN/_path/to/current/directory/, all file transfers to and from the directory 
/path/to/current/directory use the advice string /ftadv:P=WIN/. This means that all file transfers use the named profile WIN (see Section 9.4.2).

Note

Advice string parameters are case-insensitive, with the exception of file transfer profile names. For example, P=win and P=WIN point to different profiles.

Example 5

This example uses the LIKE attribute when creating an MVS PS data set with the name "userid.TEST.CPY1". The server copies RECFM, LRECL, and BLKSIZE from the PDS "COMP.DATA.CNTL".

/ftadv:like=___COMP.DATA.CNTL,T=PS/__TEST.CPY1

This example uses the LIKE attribute when creating a PDSE, "COMP.CODE". The record characteristics are copied from a PDS, "COMP.SOURCE". The PDSE is created when adding the first member, PRG1. The new PDS is estimated to need 15 MB space.

/ftadv:like=___COMP.SOURCE,T=POE,size=15000000/___COMP.CODE(PRG1)

9.4.2 File Transfer Profiles

A file transfer profile is a mechanism for pre-configuring different types of file transfers. Both the mainframe clients (scpg3, sftpg3) and the server (sshd2) use the same profile mechanism. There are two types of profiles: named profiles and filename-matched profiles.

Named profiles can be used with the file transfer advice string parameter P. A named profile provides the default values for different advice string parameters. Those default values can be overwritten with advice string parameters.

The filename-matched profiles can be used for configuring advice string default values when transferring files whose names match a certain regular expression. Again, those default values can be overwritten with advice string parameters.

File transfer profiles for Tectia server and client tools on z/OS can be set in the /opt/tec-tia/etc/ssh_ftadv_config file globally for all users and in the $HOME/.ssh2/ssh_ftadv_config file for each user separately.

The syntax of the profile is:
Each profile starts with a profile name or an `egrep` style of regular expression followed by one or more white spaces and the advice string. The profile name must start with the percent (%) character. The regular expression must not start with the % character. If the % character needs to be the first character in the regular expression, it must be escaped with a backslash (\). Note that in order to get the regular expression escape character, backslash, into a regular expression, it must be escaped by using `\\`.

The profile name and regular expression must start from the beginning of the line. There must not be any white space in front of the name or regular expression.

An advice string is a comma-separated list of name-value pairs of type name=value. There may be white spaces in the advice string. It can span over multiple lines. There must be at least one white space character in the beginning of a spanned advice string line. An advice string must not contain the file transfer advisor marker `/ftadv:/` or a file name.

Profiles with %NAME can be used with the advice string P=NAME. Profiles with REGEXP are used only for matching file names. The first REGEXP that matches a file name is used.

Comments can be added to the file with a hash (#) character. Everything on the line after # is ignored.

See the following sections for examples of Named File Transfer Profiles and Filename-Matched File Transfer Profiles in various situations.

**Examples of Named File Transfer Profiles**

**Example 1**

The following profile converts text files from Unix to MVS. ASCII is converted to EBCDIC. This profile is only used if the advice string contains the parameter P=UNIX.

```plaintext
%UNIX
    X=text,
    F=line,
    C=iso8859-1,
    D=ibm-1047
```

**Example 2**

The next profile converts text files from Windows to MVS. ASCII is converted to EBCDIC. The line delimiter is converted from the Windows style to the MVS style. This profile is only used if the advice string contains the parameter P=WIN.

```plaintext
%WIN
    X=text,
    F=line,
    C=iso8859-1,
    D=ibm-1047,
```
Example 3

The following profile can be used when transferring text files from MVS to another MVS. This profile is only used if the advice string contains the parameter P=ZOS.

```
%ZOS
  X=text,
  F=line
```

Example 4

This profile can be used when transferring text files between the Unix and MVS environments. All data sets created on the MVS side have fixed blocked lines of 80 characters per line. Again, to use this profile, P=FB80 needs to be added to the advice string.

```
%FB80
  X=text,
  F=line,
  C=iso8859-1,
  D=ibm-1047,
  O=fb,
  R=80
```

Example 5

The following profile creates record format data sets with maximum record length of 1024 bytes. The advice string P=REC is needed.

```
%REC
  F=record,
  R=1024
```

Example 6

The following profile can be used when transferring binary files.

```
%BIN
  X=bin,
  F=stream
```

Examples of Filename-Matched File Transfer Profiles

Example 1

This next profile is used whenever a data set name contains the string TXT, txt, TEXT or text in it.

```
//.*(TXT|txt|TEXT|text).*$
  X=text,
  F=line,
```
Example 2
The following profile matches files that have the extension .txt, .TXT, .C, .c, .h, .H, .log, .LOG, .conf or .CONF. Code set conversion from ASCII to EBCDIC is performed.

```
.*\.\.(txt|TXT|c|C|h|H|log|LOG|conf|CONF)$
    X=text,
    F=line,
    C=iso8859-1,
    D=ibm-1047
```

Example 3
The following profile matches binary files with extensions .gz, .Z, .tar and .bin.

```
.*\.\.(gz|Z|tar|bin)$
    X=bin,
    F=stream
```

Example 4
This profile matches all files and data set names. Data is transferred in text format and code set conversion from ASCII to EBCDIC is performed.

```
.*$
    X=text,
    F=line,
    C=iso8859-1,
    D=ibm-1047
```

Example 5
This profile specifies that record truncation is allowed. Data is transferred in text format and code set conversion from ASCII to EBCDIC is performed.

```
.*$
    X=text,
    F=line,
    U=yes,
    C=iso8859-1,
    D=ibm-1047
```

Note
With truncation, data loss may occur.
Enabling Example File Transfer Profiles

File transfer profiles are not enabled by default. 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` (globally for all users) or `$HOME/.ssh2/ssh_ftadv_config` (for a specific user).

```
> cp /opt/tectia/etc/ssh_ftadv_config.example <$HOME>/.ssh2/ssh_ftadv_config
```

9.4.3 File Transfer Environment Variables for the Server

The environment variables for file transfer on Tectia Server for IBM z/OS can be set in the `/etc/environment` file globally for all users and in the `$HOME/.ssh2/environment` file for each user separately.

File transfer server uses the following environment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH_SFTP_HOME_MVS</td>
<td>&quot;no&quot;</td>
</tr>
<tr>
<td>SSH_SFTP_RECORD_TRUNCATE</td>
<td>&quot;no&quot;</td>
</tr>
<tr>
<td>SSH_SFTP_STAGEFS_CACHE_SIZE_LIMIT</td>
<td>&quot;524288000&quot;</td>
</tr>
<tr>
<td>SSH_SFTP_STAGEFS_CACHE_ENTRY_LIFETIME</td>
<td>&quot;10&quot;</td>
</tr>
<tr>
<td>SSH_SFTP_STAGEFS_CACHE_REFRESH_INTERVAL</td>
<td>&quot;5&quot;</td>
</tr>
<tr>
<td>SSH_SFTP_DEBUG</td>
<td>NULL</td>
</tr>
<tr>
<td>SSH_SFTP_DEBUG_FILE</td>
<td>NULL</td>
</tr>
<tr>
<td>SSH_SFTP_PSEUDOVOLUME_VOLSERS</td>
<td>MIGRAT</td>
</tr>
<tr>
<td>SSH_SFTP_VOL_UNIT_MAP</td>
<td>NULL</td>
</tr>
</tbody>
</table>

If `SSH_SFTP_HOME_MVS` is set to `yes`, the file transfer server starts in the MVS side. The file transfer client sees `USER` prefix as its starting directory. Default is `no`, the file transfer server starts in the USS side. See the section called “Setting the File Transfer Home Location” for examples of using this variable.

The `SSH_SFTP_RECORD_TRUNCATE` environment variable can be used to set the default value for the `record_truncate` file transfer attribute. The valid values are `yes` and `no`. The environment variable will be overridden by a matched transfer profile or the advice string if they contain the `record_truncate` (or `U`) attribute. If none of these sources is available, `no` will be used.

The `SSH_SFTP_STAGEFS_CACHE_SIZE_LIMIT` variable specifies staging cache size in bytes. It limits the use of system resources. The default is 524288000 bytes (500 MB).

The `SSH_SFTP_STAGEFS_CACHE_ENTRY_LIFETIME` variable specifies how many seconds one cache entry is stored in the cache. After the lifetime has expired the entry is removed from the cache and system resources are released. The default is 10 seconds.

The `SSH_SFTP_STAGEFS_CACHE_REFRESH_INTERVAL` variable specifies how many seconds may pass until the cache is refreshed. The default is 5 seconds.

With `SSH_SFTP_DEBUG`, the debug level can be set for the file transfer server.

If `SSH_SFTP_DEBUG_FILE` is set, debug messages are stored in the file named in the variable.
The `SSH_SFT_PSEUDOVOLUME_VOLSERS` can be used to define a list of pseudo-volume serial numbers for data sets that are migrated or archived, see Section 6.4.3.

The environment variable `SSH_SFTP_VOL_UNIT_MAP` can be defined to provide a mapping between a volume serial number pattern and a device unit, allowing the unit to be deduced from the volume. See Section 6.3.5 for more details.

**Setting the File Transfer Home Location**

For SFTP connections, the file transfer home location is the directory on the server where the SFTP session starts. For SCP operations, the home location is the default target of the operation on the server, and directory paths are relative to the home location.

By default, Tectia Server for IBM z/OS uses the user's Unix System Services (USS) home directory as the file transfer home location.

The environment variable `SSH_SFTP_HOME_MVS` in the user's `$HOME/.ssh2/environment` file on the server can be used to control the file transfer home location.

If the environment variable is omitted or its value is no, the user's USS home directory is used as the file transfer home, for example `/u/userid/`, and the MVS user prefix must be accessed using `"/"` or `"/\"`.

If the value of the environment variable is yes, the user's MVS USERID prefix is used as the file transfer home location, for example `"/USERID.`, and the USS home directory must be accessed using `/u/userid/` or `~`.

**Examples when `SSH_SFTP_HOME_MVS`=no**

When `SSH_SFTP_HOME_MVS` is set to no (or omitted), the following `put` command run in the Windows SFTP client results to a file `/home/user1/dataset.txt` in Tectia Server for IBM z/OS:

```
sftp> open user1@zos
sftp> put dataset.txt
```

Also the following Windows SCP client command would result to the same data set:

```
$ scp pg3 --dst-site="X-TEXT" file.txt user1@zos:dataset.txt
```

The following `sput` command run in the Windows SFTP client results to a MVS sequential data set `"//USERID.MF.FILE"` in Tectia Server for IBM z/OS:

```
sftp> open user1@zos
sftp> sput remote_file //MF.FILE
```

The same applies to the Windows SCP client command.
Examples when `SSH_SFTP_HOME_MVS=yes`

When `SSH_SFTP_HOME_MVS` is set to `yes`, the following `put` command in a Windows SFTP client results to a data set `//'USER1.DATASET.TXT'` in Tectia Server for IBM z/OS:

```bash
sftp> open user1@zos
sftp> put dataset.txt
```

Also the following Windows SCP client command would result to the same data set:

```bash
$ scpg3 --dst-site="X-TEXT" file.txt user1@zos:dataset.txt
```

The following `sput` command run in the Windows SFTP client results to a USS file `~/u/user1/mf.file` in Tectia Server for IBM z/OS:

```bash
sftp> open user1@zos
sftp> sput remote_file ~/mf.file
```

Or:

```bash
sftp> open user1@zos
sftp> sput remote_file /u/user1/mf.file
```

The same applies to the Windows SCP client command.

### 9.5 Staging

The current Secure Shell protocol, as implemented in third-party products and current Tectia versions, is designed for Unix files which are byte streams. While MVS data sets are record-based, they must be staged before sending or destaged after they have been received. Staging copies a data set into byte stream format and may also convert the character set. Destaging transforms a byte stream into a file format and may also convert the character set. HFS files that do not need character set conversion, do not need to be staged.

There are two modes of staging:

- **Offline staging** stores the byte stream as an HFS file. Offline staging incurs the overhead of writing the byte stream to disk and reading it.

  Tectia client tools for z/OS support native MVS data set reading and writing and generally do not require staging.

  To use offline staging, you must run the `ssh-sft-stage` utility as a separate step. For more information, see `ssh-sft-stage(1)`.

- **Online staging** places the byte stream in virtual storage when the file is transferred. The Tectia Server for IBM z/OS (server program) will use online staging automatically if a data set cannot be transferred directly.
Online staging is more convenient and, for small files, more efficient. Online staging, for large files, will page out most of the byte stream and will page it in again. Online staging does not support files that are larger than 2 GB.

By default, staging is disabled. You can also control staging in the server configuration. When staging is disabled, all clients are required to send the data in the correct order (as consecutive blocks from beginning to end) or else the file transfer fails. In case Tectia Server for IBM z/OS detects changes in the order of the blocks, it stops the transfer and reports an error about non-serialized transfer.

To enable staging, use either one of the following methods:

- Add the staging setting for `sft-server-g3` into the `sshd2_config` configuration:
  ```
  subsystem-sftp /opt/tectia/libexec/sft-server-g3 --attribute=staging:YES
  ```

- Use the file transfer advice string in the `put` command:
  ```
  sftp> put over2gigfile.txt /ftadv:S=YES///'target'
  ```

**Note**

If the file transfer client uses the SFTP protocol for transferring the files, and does not access the files or data sets in the correct order, staging must be enabled. If the client accesses file data in random offsets, or uses checksums, staging is necessary.

**Note**

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

A summary of situations when staging is needed is shown in the following table. In the table, *Native* indicates that staging is not required, *Online* indicates that online staging is automatically used, and *Offline* indicates that the data set has to be staged offline.

<table>
<thead>
<tr>
<th>Client version</th>
<th>Server version</th>
<th>File size &lt;2GB</th>
<th>File size &gt;2GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tectia z/OS 5.2-</td>
<td>Any Unix/Windows</td>
<td>Native</td>
<td>Native</td>
</tr>
<tr>
<td>Tectia Unix/Win 5.2-</td>
<td>Tectia z/OS 5.2-</td>
<td>Native</td>
<td>Native</td>
</tr>
<tr>
<td>OpenSSH SFTP Client</td>
<td>Tectia z/OS 6.1-</td>
<td>Native</td>
<td>Native</td>
</tr>
<tr>
<td>Tectia z/OS 5.2-</td>
<td>Tectia z/OS 5.2-</td>
<td>Native</td>
<td>Native</td>
</tr>
<tr>
<td>Tectia (all) 4.x-5.1</td>
<td>Tectia z/OS 5.2-</td>
<td>Online</td>
<td>Offline</td>
</tr>
<tr>
<td>3rd-party Unix/Win</td>
<td>Tectia z/OS 5.2-6.0</td>
<td>Online</td>
<td>Offline</td>
</tr>
</tbody>
</table>
9.6 Listing Data Sets with Other SFTP Clients

Most SFTP command-line and GUI clients can be used to list data sets when Tectia Server for IBM z/OS is installed. Tectia Client supports listing data sets with sftpg3 (Tectia SFTP command-line client) on Unix and Windows.

The clients will typically contract two slashes into one. Use the alternative style of writing MVS data set names with underscores. In the examples below we show the alternative style in parentheses. See Section 9.3.1.

9.6.1 Data Set Lists

The form of data set lists depends on the file transfer client. To display data set lists, on most clients the users can use the ls and ls -l commands. Usually the ls command shows the list with relative data set names and possibly some additional information. The file size is one field in the additional information. Many SFTP clients show the file sizes of MVS data sets as "0" because Tectia Server for IBM z/OS does not determine the exact file size when listing files and does not set the size field in the protocol message. The ls -l command shows additional information formatted by the server.

When listing a PDS or PDSE, the member names are listed.

When listing a GDG, the GDG name is treated as a prefix. The listing may contain data sets that are not in the GDG index but do have data set names that have the GDG name as a prefix.

Using the ls command without any parameters displays the content of the current working directory.
Users can also define the DSN qualifier or HFS path they want to list. To use an absolute DSN, start the prefix with "/__USERID." To use a relative DSN, which will be completed with the user name, start the prefix with "/_

Some older Tectia and third-party command-line and GUI clients require that a dot sign is added after the data set prefix you want to display. This applies also to older Tectia Server for IBM z/OS file transfer clients.

For example, to list the data sets of a user's MVS home directory using absolute DSN, use the following command:

```
sftp> ls /__USERID.
```

Or using a relative DSN, use the following command:

```
sftp> ls /_.
```

### 9.6.2 Data Set Hierarchy

Tectia Server for IBM z/OS presents z/OS data sets as if they formed a hierarchy of directories where the data set name (DSN) qualifiers are the directory names.

The `cd` command can be used to change the DSN prefix. To use an absolute DSN start the prefix with "/__USER1." To use a relative DSN, which will be completed with the user name, start the prefix with "/_/.

For example, to change the working "directory" to "/__USER1", use the command:

```
sftp> cd /"USER1."
MVS prefix "USER1." is the current directory.
"USER1."
sftp>
```

If the user is connected as USER1, use the command:

```
sftp> cd //
MVS prefix `USER1.' is the current directory.
'USER1. '
sftp>
```

Once working in a "directory" ("USER1.", for example) you can go into a "subdirectory" by doing a `cd` to a qualifier:

```
sftp> cd TEST.
MVS prefix `USER1.TEST.' is the current directory.
'USER1.TEST.'
```
sftp> ls -l
"USER1.TEST.":
Volume Referred    Recfm Lrecl Blksz Dsorg    Space  Dsname
Z6SYS1 Jul 17 2006 VB    1024 27998 PS       50001  FILE
Z6SYS1 Jul 25 2006 VB    1024 27998 PS       50001  PS.FILE1
Z6SYS1 Jul 25 2006 VB    1024 27998 PS       50001  PS.FILE2
Z6SYS1 Jul 25 2006 VB    1024 27998 PS       50001  PS.FILE3

sftp> cd PS.
MVS prefix '"USER1.TEST.PS."' is the current directory.
"USER1.TEST.PS."
sftp> ls -l
"USER1.TEST.PS.":
Volume Referred    Recfm Lrecl Blksz Dsorg    Space  Dsname
Z6SYS1 Jul 25 2006 VB    1024 27998 PS       50001  FILE1
Z6SYS1 Jul 25 2006 VB    1024 27998 PS       50001  FILE2
Z6SYS1 Jul 25 2006 VB    1024 27998 PS       50001  FILE3

You can go "up" in the hierarchy with the "cd .." command:

sftp> cd ..
MVS prefix '"USER1.TEST."' is the current directory.
"USER1.TEST."
sftp> ls
"USER1.TEST."
FILE
PS.FILE1
PS.FILE2
PS.FILE3

The cd command can also be used for going into a PDS or PDSE file, for example:

sftp> cd PDS
MVS prefix '"USER1.PDS"' is the current directory.
The working directory '"USER1.PDS"' is a partitioned data set.
"USER1.PDS"
sftp> ls
"USER1.PDS."
MEM1
MEM2

Instead of using the cd command to change the directory one qualifier at a time, the whole data set name can be used, for example:
sftp> cd //USER1.TEST.PS.
MVS prefix `USER1.TEST.PS.' is the current directory.
'USER1.TEST.PS.'

9.7 Secure File Transfer Examples Using Windows and Unix Clients

Tectia Client for Windows includes a GUI and command-line applications for secure system administration and secure file transfer. Tectia Client for Unix includes only the command-line applications.

Note

When Tectia Client 6.x on Unix or Windows is used to rename or transfer files or mainframe data sets to/from z/OS hosts, and direct MVS data set access on the server side is needed, you need to define the following environment variables on the client:

- SSH_SFTP_STREAMING_MODE=ext to activate extended streaming
- SSH_SFTP_CHECKSUM_MODE=no to deactivate the checksum mode

The checksums cannot be used with streaming, because checksum calculation requires staging, but staging and streaming are mutually exclusive.

For more information, see Tectia Client 6.x User Manual.

9.7.1 File Transfers Using Windows GUI

In these examples, Tectia Client 6.2 is used. The same syntax applies to most of the Windows GUI SFTP applications.

Note that many SFTP clients show the file sizes of MVS data sets as "0". This is because Tectia Server for IBM z/OS does not determine the exact file size when listing files and does not set the size field in the protocol message.

MVS Data Set and HFS File System Hierarchies

GUI clients display MVS data sets as files and folders. Sequential and VSAM data sets are displayed as files and partitioned data sets as folders.
Partitioned data set members can be viewed by double-clicking the PDS "folder".

Figure 9.2. Viewing MVS data sets in the Tectia Secure File Transfer GUI on Windows

Figure 9.3. Viewing partitioned data sets
If a default file transfer home directory is defined as MVS, Unix System Services can be accessed by typing 
/or a Unix home directory like /home/user1) in the remote path.

Figure 9.4. Viewing USS home directory

If a default file transfer home directory is defined as USS, MVS data sets can be accessed by typing 
/__USERNAME. (for example /__USER1.) in the remote path.
Interactive File Transfers Using Windows GUI

Files and MVS data sets can be transferred with drag-and-drop either between local and remote views, or between Windows Explorer/Desktop and remote view.

These file transfers use the default file transfer profile if file transfer profiles are enabled.
Whole partitioned data sets can be transferred by transferring the PDS folders.

Specific profiles or file transfer parameters can be defined to remote path with the same format as described earlier. In the figure below, the advice string "/FTADV:P=WIN/___USER1." is used to select a Windows file transfer profile.
Unattended File Transfers Using Windows GUI

The Tectia Client GUI does not yet provide scheduled non-interactive file transfer capabilities, but the file transfer queue can be used for semi-automatic file transfers. Files, folders, and data sets can be dragged to the file transfer queue where user can select “Transfer all” with a right mouse click to initiate a batch file transfer.
9.7.2 File Transfers Using Command-Line Applications

Most of the examples in this section use Tectia Client. If not mentioned otherwise, also other clients can be used. The examples apply to both Unix and Windows command-line clients.

Interactive File Transfers Using Command-Line Applications

Tectia Client on Windows and Unix contain two file transfer clients.

- `scpg3` is a secure replacement for remote copy and provides easy secure interactive and 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 file transfer formats are the same as described earlier.

**Example 1:** Copy a Unix file to an MVS partitioned data set member using the default file transfer profile.

```bash
$ scpg3 textfile.txt user1@zos:/__USER1.TEST2.PDS/MEMBER1
```

**Example 2:** Copy an MVS sequential data set to a Windows file using a Windows file transfer profile.

```bash
C:\> scpg3 user1@zos:/ftadv:P=WIN/___USER1.TEST2.PS.FILE c:\temp\my_file.txt
```
**Example 3:** Copy a partitioned data set with members to Unix a directory and files using the default file transfer profile.

```
$ scp3 -r user1@zos:/_TEST2.PDS /tmp/pds/
```

**Example 4:** File transfers between Unix and z/OS using `sftp3`.

```
$ sftp3 user1@zos

user1@zos's password:

sftp> sget /_TEST2.PDS/MEMBER1 /tmp/member1

sftp> sget /FTADV:X=BIN/__TEST.PS.FILE1

sftp> sput /tmp/local_file.txt /FTADV:P=FB80/___SYSUID..SSZ.JCL1
```

1. Connect from Unix to z/OS using `sftp3`.
2. Fetch an MVS PDS member into a Unix file.
3. Fetch an MVS data set. Use a file transfer advice string to set `TRANSFER_MODE` (x) to binary.

Note that, by default, a sequential data set - not a partitioned data set - is created. If you want to store your source file to a PDS, you can do it in one of the following ways:

- First create the PDS using the `mkdir` command. After this you can put your source file to the PDS, giving a valid member name if the source file name is not suitable.
- Specify the directory size (`DIRECTORY_SIZE`) in the file transfer advice string as follows:

  ```
  sftp> sput /tmp/local_file.txt /FTADV:P=FB80,M=10/___SYSUID..SSZ.JCL1
  ```

**Example 5:** Using the `sftp3` `site` command to define additional data set parameters.

**Note**

The `sftp3` `site` commands are recognized only by Tectia products. File transfer advice strings should be used with other clients than Tectia.

Parameters can be entered either one by one, or several parameters can be delimited by commas or spaces. Both long parameters and abbreviations can be used. The plain `site` command outputs the list of entered parameters.

```
sftp> site LRECL=80
sftp> site RECFM=FB,VOLUMES=TEST
sftp> site X=TEXT C=ISO8859-1 D=IBM-1047
sftp> site
LRECL=80
RECFM=FB
TRANSFER_CODESET=ISO8859-1
TRANSFER_FILE_CODESET=IBM-1047
```

Tectia® Server 6.6 for IBM z/OS User Manual © 2007–2020 SSH Communications Security Corporation
TRANSFER_MODE=TEXT
VOLUMES=TEST

1. Set record length (LRECL).
2. Set record format (RECFM) and volumes (VOLUMES), separated by a comma.
3. Set TRANSFER_MODE | x, TRANSFER_CODESET | c and TRANSFER_FILE_CODESET | d, separated by spaces.
4. List the currently set site parameters.

Example 6: File transfers using Unix OpenSSH client.

```bash
$ sftp user1@zos
Connecting to zos
user1@zos's password:
sftp> get /_TEST2.PDS/MEMBER1 /tmp/member1
sftp> get /FTADV:X=BIN/__TEST.PS.FILE1
sftp> put /tmp/local_file.txt /FTADV:P=FB80/___&SYSUID..SSZ.JCL1
```

Note that, by default, a sequential data set - not a partitioned data set - is created. To store your source file to a PDS, follow the instructions presented in Example 4 above.

Unattended File Transfers Using Command-Line Applications

`scp` commands can be used as Scheduled Tasks on Windows, or as cron jobs on Unix. With `sftp`, batch mode can be used. The syntaxes for `scp` and `sftp` are the same as described before.

9.7.3 File Transfers Using FTP-SFTP Conversion

Tectia ConnectSecure provides an FTP-SFTP conversion functionality that enables users to use their existing interactive and unattended FTP applications and tools. FTP-SFTP conversion can also be used for file transfers from Windows and Unix clients to z/OS servers.

For configuring FTP-SFTP Conversion for Windows or Unix, see the Tectia ConnectSecure Administrator Manual at https://www.ssh.com/manuals/. Once configured, existing FTP application can be used like before.

Example: Using FTP-SFTP conversion to transfer MVS data set using existing FTP client application. All the standard FTP commands can be used.

```bash
$ ftp zos
Connected to zos (10.10.10.10).
220----- SSH FTP-SFTP Conversion ----- 220-----------------------------
220 Your FTP connection is now SECURED!
Name (zos:user1): user1
331 Send password please.
Password:
230 You are logged on.
Remote system type is MVS.
ftp> ls
```
Entering Passive Mode (127,0,0,1,78,32).
Connecting to data port.

<table>
<thead>
<tr>
<th>Volume Referred</th>
<th>Recfm</th>
<th>Lrecl</th>
<th>BlkSz</th>
<th>Dsorg</th>
<th>Space</th>
<th>Dsname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z6SYS1 Jul 17 2006 VB</td>
<td>1024</td>
<td>27998 PS</td>
<td>50001</td>
<td>BINARY.FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z6SYS1 Jul 17 2006 VB</td>
<td>1024</td>
<td>27998 PS</td>
<td>50001</td>
<td>FILE1.PS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z6SYS1 Jul 17 2006 FB</td>
<td>80</td>
<td>6080 PS</td>
<td>50001</td>
<td>FILE1.VSAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z6SYS1 Jul 17 2006 FB</td>
<td>80</td>
<td>27920 PO</td>
<td>50001</td>
<td>PDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z6SYS1 Jul 25 2006 VB</td>
<td>1024</td>
<td>27998 PO</td>
<td>50001</td>
<td>SAMPLIB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z6SYS1 Jul 25 2006 VB</td>
<td>1024</td>
<td>27998 PS</td>
<td>50001</td>
<td>TEST.PS.FILE2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z6SYS1 Jul 25 2006 VB</td>
<td>1024</td>
<td>27998 PO</td>
<td>50001</td>
<td>TEST2.PDS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

List completed successfully.

ftp> get FILE1.PS /tmp/file1.txt
local: /tmp/file1.txt remote: FILE1.PS
Entering Passive Mode (127,0,0,1,78,33).
Connecting to data port.
Transfer completed successfully.
49 bytes received in 0.421 secs (0.11 Kbytes/sec)

ftp> cd test2
"USER1.TEST2." is the working directory name prefix.

ftp> cd PDS
The working directory "USER1.TEST2.PDS" is a partitioned data set.

ftp> ls
List completed successfully.

ftp> put textfile.txt member2
local: textfile.txt remote: member2
Entering Passive Mode (127,0,0,1,78,34).
Connecting to data port.
Transfer completed successfully.
71 bytes sent in 3.3e-05 secs (2.1e+03 Kbytes/sec)

Example 2: Using the site command to define additional data set parameters (on Tectia Client 5.2 and later).

ftp> site
(arguments to SITE command) LRECL=200
SITE command was accepted

ftp> site recfm=FB,C=ISO8859-1,D=IBM-1047,X=TEXT
SITE command was accepted

ftp> rstatus
SSH FTP-SFTP conversion status:
  Version 5.2 Build 53
  Connected to 10.1.49.158
211-Logged in as user1
211-Site parameters: O=FB,R=200,X=TEXT,C=ISO8859-1,D=IBM-1047
211 End of status.

ftp> put textfile.txt //USER1.SITE.TEST
local: textfile.txt remote: //USER1.SITE.TEST
227 Entering Passive Mode (127,0,0,1,78,38).
150 Connecting to data port.
226 Transfer completed successfully.
71 bytes sent in 4.1e-05 secs (1.7e+03 Kbytes/sec)
ftp>
Appendix A Connection Broker and SOCKS Proxy Configuration Files

The authentication and connection profile settings for Tectia client tools for z/OS are made in the Connection Broker configuration, because the Connection Broker handles all cryptographic operations and authentication-related tasks for Tectia client tools for z/OS.

This section describes the configuration settings of the Tectia client components on z/OS. The configuration settings are made in an XML file that follows the given DTD.

A.1 Configuration File

The elements of the XML-based configuration files ssh-broker-config.xml and ssh-socks-proxy-config.xml are described in ssh-broker-config(5).
**ssh-broker-config**

*ssh-broker-config* -- Tectia Connection Broker configuration file format

The Connection Broker configuration file *ssh-broker-config.xml* is used by Tectia Client and ConnectSecure on Unix and Windows, and additionally by the Tectia client tools on IBM z/OS and Linux for IBM System z. The Connection Broker configuration file must be a valid XML file that follows the *ssh-broker-ng-config-1.dtd* document type definition.

The SOCKS Proxy configuration file *ssh-socks-proxy-config.xml* is used by the Tectia SOCKS Proxy on z/OS. It must be a valid XML file that follows the *ssh-broker-ng-config-1.dtd* document type definition.

**Connection Broker Files**

The Connection Broker reads three configuration files (if all are available):

1. **The** *ssh-broker-config-default.xml* file **is read first. It holds the factory default settings. It is not recommended to edit the file, but you can use it to view the default settings.**

   This file must be available and correctly formatted for the Connection Broker to start.

2. **Next, the Connection Broker reads the global configuration file. The settings in the global configuration file override the default settings.**

   If the global configuration file is missing or malformed, the Connection Broker will start normally, and will read the user-specific configuration file, instead. A malformed global configuration file is ignored and the default settings or user-specific settings, if they exist, are used instead.

3. **Last, the Connection Broker reads the user-specific configuration file, if it is available. The settings in the user-specific configuration file override the settings in the global configuration file, with the following exceptions:**

   - The following settings from the user-specific configuration are combined with the settings of the global configuration file:
     - In *general* element, the *key-stores, cert-validation and file-access-control settings*
     - In *profiles* element, all settings
     - In *static-tunnels* element, all settings.
   - If a connection profile with the same name has been defined in both the global configuration file and user-specific configuration file, the latter one is used.
   - If the *filter-engine* settings have been defined in the global configuration file, and the file is valid (not malformed), those settings are used, and any *filter-engine* settings made in the user-specific configuration file are ignored.
If the user-specific configuration file is missing, the Connection Broker will start using the previously read configuration files. However, if a user-specific configuration exists but is malformed, the Connection Broker will not start at all.

On z/OS, the default configuration file locations are as follows:

- the default configuration:
  
  /opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-config-default.xml

- the global configuration:
  
  /opt/tectia/etc/ssh-broker-config.xml

- the user-specific configuration:
  
  $HOME/.ssh2/ssh-broker-config.xml

- the XML DTD:
  
  /opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd

The following sections describe the options available in the Connection Broker configuration file. For more information on the syntax of the configuration file, see Section A.3.

**SOCKS Proxy Files**

The SOCKS Proxy also reads three configuration files (if all are available). The files are read in the same order as with the Connection Broker. See the section called “Connection Broker Files” above.

On z/OS, the default configuration file locations are as follows:

- the default configuration:

  /opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-socks-proxy-config-default.xml

- the global configuration:

  /opt/tectia/etc/ssh-socks-proxy-config.xml

- the user-specific configuration:

  $HOME/.ssh2/ssh-socks-proxy-config.xml

- the XML DTD:

  /opt/tectia/etc/ssh-tectia/auxdata/ssh-broker-ng/ssh-broker-ng-config-1.dtd

**Note**

We recommend you to use either the global or the user-specific configuration file, but not both. If the user-specific configuration file is to be used, please remove the global configuration file. This is to ensure that no settings in the user-specific configuration file will be unexpectedly overridden by settings in the global configuration file.
The information in the following sections on the options of the Connection Broker configuration file is also valid for the options of the SOCKS Proxy configuration file. For more information on the syntax of the configuration file, see Section A.3.

Environment Variables

Two kinds of environment variables can be used in the Connection Broker configuration file. In addition to the system-level environment variables, you can use special variables that are Tectia specific. The environment variables take precedence over the special variables. So if an environment variable and a special variable have the same name, the environment variable will be used.

All alphanumeric characters and the underscore ‘_’ sign are allowed in environment variables. The variable name ends to the first character that is not allowed.

You can define for example file or directory paths with environment variables, and they will be expanded to their values as explained below.

%VARIABLENAME%
   Replaced with the value of the environment variable if one has been defined. The variable is matched case-insensitively. If the variable is not defined, the string ‘%VARIABLENAME%’ is the result.

$VARIABLENAME
   Replaced with the value of the environment variable if one has been defined. The variable is matched case-sensitively on Unix and case-insensitively on Windows. If the variable is not defined, it is replaced with an empty string.

${VARIABLENAME:text}
   Replaced with the value defined for ’$VARIABLENAME’ with the ‘text’ appended to it.

${VARIABLENAME:default_value}
   Replaced with the value defined for ’$VARIABLENAME’, or replaced with the ‘default_value’ if the variable is not set.

The Tectia specific special variables are:

%U or %username%
   Replaced with the currently logged in user name.

%username-without-domain%
   Replaced with the currently logged in user name in short format, i.e. without the domain part. Available on Windows.

%G or %groupname%
   Replaced with the group name of the currently logged in user.

%D or %homedir%
   Replaced with the home directory defined for the currently logged in user.
%IU or %userid%
  Replaced with the user identifier defined for the currently logged in user.

%IG or %groupid%
  Replaced with the group identifier defined for the currently logged in user.

The special variables can also be entered using the Unix format, for example, $username.

**Document Type Declaration and the Root Element**

The Connection Broker configuration file is a valid XML file and starts with the Document Type Declaration.

The root element in the configuration file is secsh-broker. It can include general, default-settings, profiles, static-tunnels, gui, filter-engine, and logging elements.

An example of an empty configuration file is shown below:

```
<!DOCTYPE secsh-broker SYSTEM "ssh-broker-ng-config-1.dtd">
<secsh-broker version="1.0">
  <general />
  <default-settings />
  <profiles />
  <static-tunnels />
  <gui />
  <filter-engine />
  <logging />
</secsh-broker>
```

**The general Element**

The general element contains settings such as the cryptographic library and the key stores to be used.

The general element can contain zero or one instance of the following elements: crypto-lib, cert-validation, key-stores, user-config-directory, protocol-parameters; and multiple known-hosts elements.

**crypto-lib**

This element selects the cryptographic library mode to be used. Either the standard version (standard) or the FIPS 140-2 certified version (fips) of the cryptographic library can be used. The library name is given as a value of the mode attribute. By default, standard cryptographic libraries are used. The OpenSSL cryptographic library is used in the *FIPS mode*.

```
<crypto-lib mode="standard" />
```

**Note**

The FIPS library is not available on z/OS.
For a list of platforms on which the FIPS library has been validated or tested, see *Tectia Client/Server Product Description*.

**cert-validation**

This element defines public-key infrastructure (PKI) settings used for validating remote server authentication certificates. The element can have the following attributes: `end-point-identity-check`, `default-domain`, `http-proxy-url`, `socks-server-url` and `max-path-length`.

The `end-point-identity-check` attribute specifies whether the client will verify the server's host name or IP address against the Subject Name or Subject Alternative Name (DNS Address) specified in the server host certificate. The default value is `yes`. If set to `no`, the fields in the server host certificate are not verified and the certificate is accepted based on the validity period and CRL check only.

**Caution**

Setting `end-point-identity-check="no"` is a security risk. Then anyone with a certificate issued by the same trusted certification authority (CA) that issues the server host certificates can perform a man-in-the-middle attack on the server.

Alternatively, if set to `ask`, the user can decide to either cancel or continue establishing the connection in case that the server's host name does not match the one in the certificate.

The `default-domain` attribute can be used when the end-point identity check is enabled. It specifies the default domain part of the remote system name and it is used if only the base part of the system name is available. The `default-domain` is appended to the system name if it does not contain a dot (.).

If the default domain is not specified, the end-point identity check fails, for example, when a user tries to connect to a host "rock" giving only the short host name and the certificate contains the full DNS address "rock.example.com".

The `http-proxy-url` attribute defines an HTTP proxy and the `socks-server-url` attribute defines a SOCKS server for making LDAP or OCSP queries for certificate validity.

The address of the server is given as the value of the attribute. The format of the address is `socks://username@socks_server:port/network/netmask, network/netmask ...` (with a SOCKS server) or `http://username@proxy_server:port/network/netmask, network/netmask ...` (with an HTTP proxy).

For example, to make the SOCKS server use host `socks.ssh.com` and port `1080` for connections outside of networks `192.196.0.0` (16-bit domain) and `10.100.23.0` (8-bit domain), and to get these networks connected directly, set `socks-server-url` as follows:

```plaintext
*socks://mylogin@socks.ssh.com:1080/192.196.0.0/16,10.100.23.0/24*
```

The `max-path-length` attribute limits the length of the certification paths when validating certificates. It can be used to safeguard the paths or to optimize against the paths getting too long in a deeply hierarch-
ical PKI or when the PKI is heavily cross-certified with other PKIs. Using the attributes requires knowing the upper limit of the paths used in certificate validation. For example:

```xml
<cert-validation max-path-length="6">
  <ldap-server address="ldap://myldap.com" port="389" />
  <dod-pki enable="yes" />
  <ca-certificate name="CA 1" file="ca-certificate1.crt" />
</cert-validation>
```

In the example, the path is limited to six certificates, including the end-entity and root CA certificates. If not specified, the default value is 10. Decrease the value to optimize the validation if the maximum length of the encountered paths in the certificate validation is known.

The `cert-validation` element can contain multiple `ldap-server`, `ocsp-responder`, `crl-prefetch` elements, one `dod-pki` element, and multiple `ca-certificate` and `key-store` elements. The elements have to be in the listed order.

**ldap-server**

This element specifies an LDAP server `address` and `port` used for fetching CRLs and/or subordinate CA certificates based on the issuer name of the certificate being validated. Several LDAP servers can be specified by using several `ldap-server` elements.

CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be verified if the point exists.

The default value for `port` is 389.

**ocsp-responder**

This element specifies an OCSP (Online Certificate Status Protocol) responder service address in URL format with attribute `url`. Several OCSP responders can be specified by using several `ocsp-responder` elements.

If the certificate has a valid Authority Info Access extension with an OCSP Responder URL, it will be used instead of this setting. Note that for the OCSP validation to succeed, both the end-entity certificate and the OCSP Responder certificate must be issued by the same CA.

The `validity-period` (in seconds) can be optionally defined. During this time, new OCSP queries for the same certificate are not made but the old result is used. The default validity period is 0 (a new query is made every time).

**crl-prefetch**

This element instructs Tectia client tools for z/OS to periodically download a CRL from the specified URL. The `url` value can be an LDAP or HTTP URL, or it can refer to a local file. The file format must be either binary DER or base64, PEM is not supported.
To download CRLs from the local file system, define the file URL in this format:

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

To download CRLs from an LDAP server, define the LDAP URL in this format:

```
ldap://ldap.server.com:389/CN=Root%20CA,
   OU=certification%20authorities,DC=company,
   DC=com?certificaterevocationlist
```

Use the `interval` attribute to specify how often the CRL is downloaded. The default is 3600 seconds.

**dod-pki**

This element defines whether the certificates are required to be compliant with the US Department of Defense Public-Key Infrastructure (DoD PKI). In practice, this means that the Digital Signature bit must be set in the Key Usage of the certificate. The `enable` attribute can have a value of `yes` or `no`. The default is `no`.

**ca-certificate**

This element defines a certification authority (CA) used in server authentication. It can have four attributes: `name`, `file`, `disable-crls`, and `use-expired-crls`.

The `name` attribute must contain the name of the CA.

The element must either contain the path to the X.509 CA certificate file as a value of the `file` attribute, or include the certificate as a base64-encoded ASCII block.

CRL checking can be disabled by setting the `disable-crls` attribute to `yes`. The default is `no`.

Expired CRLs can be used by setting a numeric value (in seconds) for the `use-expired-crls` attribute. The default is 0 (do not use expired CRLs).

**key-store**

This element defines CA certificates stored in an external key store for server authentication. Currently it is used only on z/OS for CA certificates stored in System Authorization Facility (SAF).

The `cert-validation/key-store` element has four attributes: `type`, `init`, `disable-crls`, and `use-expired-crls`.

The `type` attribute defines the key store type. Currently the only supported type is "zos-saf".

The `init` attribute is the key-store-provider-specific initialization info.

CRL checking can be disabled by setting the `disable-crls` attribute to `yes`. The default is `no`.

Expired CRLs can be used by setting a numeric value (in seconds) for the `use-expired-crls` attribute. The default is 0 (do not use expired CRLs).
For key store configuration examples, see the section called “Key Store Configuration Examples”.

An example of a certificate validation configuration is shown below:

```xml
<cert-validation end-point-identity-check="yes"
    default-domain="example.com"
    http-proxy-url="http://proxy.example.com:8080"/>
<ldap-server address="ldap://ldap.example.com:389" />
<ocsp-responder url="http://ocsp.example.com:8090"
    validity-period="0" />
<crl-prefetch url="file:///full.path.to.crlfile"
    interval="1800" />
<dod-pki enable="no" />
<ca-certificate name="ssh_ca1"
    file="ssh_ca1.crt"
    disable-crls="no"
    use-expired-crls="100" />
</cert-validation>
```

**key-stores**

This element defines settings for user public-key and certificate authentication.

Under the `<general>` element, there can be one `<key-stores>` instance which in turn can have any number of `<key-store>`, `<user-keys>`, and `<identification>` elements, and the order of the elements is free.

Special variables and environment variables can be used when defining the values for the elements. The following variables can be used and they will be expanded as follows:

- `%U` = `%USERNAME%` = user name
- `%USERNAME-WITHOUT-DOMAIN%` = user name without the domain part
- `%IU` = `%USERID%` = user ID
- `%IG` = `%GROUPID%` = user group ID
- `%D` = `%HOMEDIR%` = the user's home directory
- `%G` = `%GROUPNAME%` = the name of the user's default group

Also environment variables are replaced with their current values. For example it is possible to use strings `$HOME` or `%HOME%` to expand to user's home directory (if environment variable `HOME` is set).

**Note**

Short alias names (for example, `%U`) are case-sensitive and long alias names (for example, `%USERNAME%`) are case-insensitive.
key-store

Each of the **key-store** elements configures one key store provider. The **key-stores/key-store** element can take the following attributes: **type** and **init**.

The **type** attribute is the key store type. The currently supported types are "mscap", "pkcs11", "software", and "zos-saf".

The **init** attribute is the initialization info specific to the key-store-provider. The initialization string can contain special strings explained above in **key-stores**.

For key store configuration examples, see the section called “Key Store Configuration Examples”.

user-keys

The **user-keys** element can be used to override the default directory for the user keys. The **user-keys** element can take the following attributes:

The **directory** attribute defines the directory where the user private keys are stored. Enter the full path.

The **passphrase-timeout** attribute defines the time (in seconds) after which the passphrase-protected private key will time out, and the user must enter the passphrase again. The default is 0, meaning that the passphrase does not time out. The value of this element should be longer than the **passphrase-idle-timeout** value.

By default, the Connection Broker keeps the passphrase-protected private keys open once the user has entered the passphrase successfully. This can be changed with the passphrase timeout options. When **passphrase-timeout** is set, the private key stays open (usable without further passphrase prompts) until the timeout expires. The **passphrase-timeout** attribute sets the hard timeout, that is set only once when the key is opened and will not be reset even if the key is used multiple times.

The **passphrase-idle-timeout** attribute defines the time (in seconds) after which the passphrase-protected private key will time out unless the user accesses or uses the key. The **passphrase-idle-timeout** is reset every time the key is accessed. The default is 0, meaning that the passphrase never times out.

Both of the timeout options can be set simultaneously, but notice that if the idle timeout is set longer than the hard timeout, the idle timeout has no effect.

identification

The **identification** element can be used to override the default location of the identification file that defines the user keys. The **identification** element can take the following attributes:

The **file** attribute specifies the location of the identification file. Enter the full path.
The `base-path` attribute defines the directory where the identification file expects the user private keys to be stored. This element can be used to override the default relative path interpretation of the identification file (paths relative to the identification file directory).

The `passphrase-timeout` attribute defines the time (in seconds) after which the user must enter the passphrase again. The default is 0, meaning that the passphrase is not re-requested.

The `passphrase-idle-timeout` attribute defines a time (in seconds) after which the passphrase times out if there are no user actions. The default is 0, meaning that the passphrase does not time out.

The timeout settings affect only those private keys that are listed in the identification file.

**strict-host-key-checking**

**Note**

This element is deprecated starting from Tectia client tools for z/OS version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `host-key-always-ask` and `accept-unknown-host-keys` options. See `auth-server-publickey` for details.

**host-key-always-ask**

**Note**

This element is deprecated starting from Tectia client tools for z/OS version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on the values of this option and the `strict-host-key-checking` and `accept-unknown-host-keys` options. See `auth-server-publickey` for details.

**accept-unknown-host-keys**

**Note**

This element is deprecated starting from Tectia client tools for z/OS version 6.1.4.

This element is supported in configuration for backwards compatibility and used only if the `policy` attribute of the `server-authentication-methods/auth-server-publickey` element under `default-settings` or `profiles/profile` is not defined. In this case, the host key policy is interpreted based on
the values of this option and the strict-host-key-checking and host-key-always-ask options. See auth-server-publickey for details.

⚠️ **Caution**

Consider carefully before enabling this option. Disabling the host-key checks makes you vulnerable to man-in-the-middle attacks.

**user-config-directory**

This element can be used to change the storage location of the user-specific configuration files away from the default which is $HOME/.ssh2/ on Unix, and "%APPDATA%\SSH" on Windows. It can be used for example, if you want to store all client-side configurations to a centralized location.

When this element is added to the global configuration file, the Connection Broker reads the following user-specific files in the defined location:

- User's key file
- User's own configuration files
- User's known host keys
- User's random_seed file
- Windows GUI profile files: 1.ssh2, 2.ssh2
- The startup batch file for the **sftpg3** client: ssh_sftp_batch_file

🔍 **Note**

Stop all existing SSH applications before modifying the **user-config-directory** setting in the Connection Broker configuration.

The **user-config-directory** setting affects all Tectia products running on the same host, for example Tectia Client and Tectia ConnectSecure.

The **user-config-directory** option takes an attribute **path**, whose value can be either a directory path or one of the following variables:

- %U: The user name.
- %username%: The user name.
- %username-without-domain%: The user name without domain definition.
- %D: The user's home directory.
- %homedir%: The user's home directory.
• `%USER_CONFIG_DIRECTORY%`: The user-specific configuration directory.

• `%IU`: The user's ID, on Unix only

• `%userid%`: The user's ID, on Unix only

• `%IG`: The group ID, on Unix only

• `%groupid%`: The group ID, on Unix only

The default is `%USER_CONFIG_DIRECTORY%`. This variable refers to the user-specific configuration directory: `$HOME/.ssh2` on Unix, and `%APPDATA%\SSH` on Windows. The `%USER_CONFIG_DIRECTORY%` variable cannot be used in other settings.

file-access-control

On Unix, this element can be used to enable checking of file access permissions defined for the global and user-specific configuration files, and for the private keys files. If the permissions are not as expected, the Connection Broker will refuse to start, or to use certain private keys.

By default this setting is disabled. On Windows, this element has no effect.

The file permissions are checked differently, if the `file-access-control` element is set in both the global and user configuration files, or just in one of them.

In the table: "no" means `file-access-control enable="no"`. The "-" sign means that the setting is not defined in the file at all.

When the file access permissions are checked, the controls are applied as follows:

• Expected permissions for the global configuration file: read rights for all, write rights only for the user and group. If the permissions are any wider, the Connection Broker will not start.

• Expected permissions for the user configuration file: only the user has read and write rights. If the permissions are any wider, the Connection Broker will not start.

• Expected permissions for the private key files: only the user has read and write rights. If the permissions are any wider, keys that do not pass the check will be ignored.

protocol-parameters

This element contains protocol-specific values that can be used to tune the performance. It should be used only in very specific environments. In normal situations the default values should be used.

The `threads` attribute can be used to define the number of threads the protocol library uses (fast path dispatcher threads). This attribute can be used to allow more concurrent cryptographic transforms in the protocol on systems with more than four CPUs. If the value is set to zero, the default value is used.

Example of the `threads` attribute:
known-hosts

This element can be used to specify locations for storing the host keys of known server hosts, and to define the storage format of the host key files. If no known-hosts directories are specified, the known host keys are stored to the default directories. See the section called “Files” for the default locations. On z/OS (only), this element can contain key-store elements.

This element can be used:

• To specify non-default directories that contain the public-key data or public-key files of known server hosts.

• To specify a non-default location for OpenSSH-style known_hosts files that contain the public-key data of known server hosts.

• (On z/OS) To specify a SAF key store that contains the certificates of known server hosts.

The server host keys are searched in the known-hosts paths in the order they are specified in the configuration. The settings of the last defined known-hosts element are used when storing new host keys.

If you define any known-hosts file settings, the default OpenSSH files will be overridden. So if you wish to make the Connection Broker use both the default OpenSSH locations and other locations specified in the configuration, you need to specify all the locations separately.

You can define several known-hosts elements, and each of them can contain one or several attributes: path, directory, file and filename-format.

The path attribute requires a full path to the known-hosts file or directory as the value. For example:

```xml
<known-hosts path="/u/username/.ssh/known_hosts" />
<known-hosts path="/etc/ssh2/hostkeys" />
<known-hosts path="/u/username/.ssh2/hostkeys" />
<known-hosts path="/h/username/hostkeys" filename-format="plain" />
```

The directory attribute is used to define that known host keys are saved to a non-default directory. Enter the complete path to the directory as the value. If the defined directory does not exist, it will be created during the first connection attempt. If a file is found in its place, the connection will be made but the host key will not be stored, and the user gets a warning about it. The filename-format attribute can be used together with the directory setting to define in which format the host key files will be stored. Example of the directory attribute:

```xml
<known-hosts directory="<path_to_dir>/MyKEYS" filename-format="plain" />
```

The path or directory (whichever is present) defined in the last known-hosts element in the configuration file will be used when storing new known host keys. If both attributes are present in the last known-hosts element, the location specified in the directory attribute will be used.
The file attribute is used to point to an OpenSSH-style known_hosts file. Enter the complete path to the file as the value. If a directory is found in its place, it is considered an error, and the connection attempt will fail. In case the known-hosts element only contains the file attribute, and the defined OpenSSH known_hosts file exists, the received host keys are searched first in the defined file, and if not found there, the search continues in the default Tectia-specific locations.

Example of the file attribute:

```
<known-hosts file="<path_to_file>/ssh2/openSSH_keys" />
```

An empty file or path attribute will disable the handling of the OpenSSH known_hosts file:

```
<known-hosts file="" />
```

or

```
<known-hosts path="" />
```

The filename-format attribute defines the format in which new host key files are stored. The filename-format attribute is only relevant for the last specified known-hosts element and for the default directory.

The filename-format attribute takes the values: hash (default), plain, and default (equals to hash).

With value hash, the host key files will be stored in format: keys_<hash>, for example "keys_182166d2efe5a134d3fb948646e0b48f780bff6c".

With value plain, the file name format will be key_<port>_<hostname>.pub, where <port> is the port the Secure Shell server is running on and <hostname> is the host name you use when connecting to the server; for example "key_22_my.example.com.pub".

Setting `<known-hosts filename-format="plain" />` changes the storage format of host key files for the next known-hosts elements or for the default storage location if no other known-hosts elements are present.

The filename-format="default" alternative can be used as the last option when the same known-hosts element is used to define several locations for the host keys some of which store the keys in plain format.

For more information on the host key storage formats, see Section 4.2.1.

**key-store**

This element defines an external key store for certificates of known server hosts. Currently it is used only on z/OS for server certificates stored in System Authorization Facility (SAF).

The known-hosts/key-store element has two attributes: type and init.

The type attribute defines the key store type. Currently the only supported type is "zos-saf".

The init attribute is the key-store-provider-specific initialization info.

For key store configuration examples, see the section called “Key Store Configuration Examples”.

© 2007–2020 SSH Communications Security Corporation
extended

This element is reserved for future use.

Key Store Configuration Examples

Example with Software Provider

The software provider handles key pairs stored on disk in standard Secure Shell v2 or legacy OpenSSH formats and X.509 certificates stored in native X.509, PKCS #7, and PKCS #12 formats.

To add a single key file (for example, /u/exa/keys/enigma and /etc/my_key), specify both the private key file and the public key file:

```xml
<key-stores>
  <key-store type="software"
    init="key_files(/u/exa/keys/enigma.pub,/u/exa/keys/enigma)" />
  <key-store type="software"
    init="key_files(/etc/my_key.pub,/etc/my_key)" />
</key-stores>
```

To add all keys from a specific directory (for example all keys from /u/exa/keys and /etc/keys):

```xml
<key-stores>
  <key-store type="software"
    init="directory(path(/u/exa/keys))" />
  <key-store type="software"
    init="directory(path(/etc/keys))" />
</key-stores>
```

Example with z/OS SAF Provider

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]})... [TRUST-ANCHORS]}
```

KEYS( . . . ) may repeat. The subattributes 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.
TRUST-ANCHORS

Specifies that the certificates in the key ring are trusted CA certificates.

Specifying CA certificates stored in SAF as trusted for server authentication:

```xml
<cert-validation>
  ...
  <key-store type="zos-saf"
    init="KEYS(ID(SSHD2) RING(SSH-HOSTCA)) TRUST-ANCHORS"
    disable-crls="no"
    use-expired-crls="100" />
</cert-validation>
```

Specifying server certificates stored in SAF for server authentication:

```xml
<known-hosts>
  ...
  <key-store type="zos-saf"
    init="KEYS(ID(USER) RING(SSH-HOSTKEYS))" />
</known-hosts>
```

Specifying user certificates stored in SAF for user authentication:

```xml
<key-stores>
  <key-store type="zos-saf"
    init="KEYS(ID(%U) RING(%U))" />
</key-stores>
```

The **default-settings** Element

The **default-settings** element defines the default connection-related settings. Profile-specific settings can override these settings. See the section called "The profiles Element".

The **default-settings** element can contain zero or one instance of the following elements in the listed order: ciphers, macs, kexs, hostkey-algorithms, rekey, authentication-methods, hostbased-default-domain, compression, proxy, idle-timeout, tcp-connect-timeout, keepalive-interval, exclusive-connection, server-banners, forwards, extended, remote-environment, server-authentication-methods, authentication-success-message, sftp-g3-mode, terminal-selection, terminal-bell, close-window-on-disconnect, quiet-mode, checksum, and address-family.

The **default-settings** element can take one attribute:

The **user** attribute can be used to define a default user name to be used when connecting to remote servers. The value of the **user** attribute can be one of the following:

- A generic user name that will be used in connections unless another user name is specified in the connection profile settings or in the connection attempt. Note that the user name is treated case sensitively.
- "%USERNAME%" can be used to apply the user name of the currently logged in user.
• In case this option is used but left empty, the Connection Broker will prompt the user for a user name.

The **default-settings** element can contain the following elements:

**ciphers**

This element defines the ciphers that the client will propose to the server. The **ciphers** element can contain multiple **cipher** elements.

The ciphers are tried in the order they are specified.

**cipher**

This element selects a cipher **name** that the client requests for data encryption.

The supported ciphers are:

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Cipher</th>
<th>Cipher</th>
</tr>
</thead>
<tbody>
<tr>
<td>3des-cbc</td>
<td>aes256-ctr</td>
<td>twofish256-cbc</td>
</tr>
<tr>
<td>aes128-cbc</td>
<td>arcfour</td>
<td><a href="mailto:crypticore128@ssh.com">crypticore128@ssh.com</a></td>
</tr>
<tr>
<td>aes192-cbc</td>
<td>blowfish-cbc</td>
<td><a href="mailto:seed-cbc@ssh.com">seed-cbc@ssh.com</a></td>
</tr>
<tr>
<td>aes256-cbc</td>
<td>twofish-cbc</td>
<td>none (no encryption)</td>
</tr>
<tr>
<td>aes128-ctr</td>
<td>twofish128-cbc</td>
<td></td>
</tr>
<tr>
<td>aes192-ctr</td>
<td>twofish192-cbc</td>
<td></td>
</tr>
</tbody>
</table>

The default ciphers used by the Connection Broker are, in order: **aes128-cbc**, **aes128-ctr**, **aes192-cbc**, **aes192-ctr**, **aes256-cbc**, **aes256-ctr**, and **3des-cbc**.

```xml
<ciphers>
  <cipher name="aes128-cbc" />
  <cipher name="3des-cbc" />
</ciphers>
```

**macs**

This element defines the MACs that the client will propose to the server. The **macs** element can contain multiple **mac** elements.

The MACs are tried in the order they are specified.

**mac**

This element selects a MAC **name** that the client requests for data integrity verification.

The supported MAC algorithms are:

<table>
<thead>
<tr>
<th>MAC</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmac-md5</td>
<td><a href="mailto:hmac-sha256@ssh.com">hmac-sha256@ssh.com</a></td>
</tr>
<tr>
<td>hmac-md5-96</td>
<td><a href="mailto:hmac-sha384@ssh.com">hmac-sha384@ssh.com</a></td>
</tr>
</tbody>
</table>

1. crypticore128@ssh.com is not supported on Tectia Server for Linux on IBM System z.
The default MACs used by the Connection Broker are, in order:

```
hmac-sha1
hmac-sha1-96
hmac-sha2-256
hmac-sha256-2@ssh.com
hmac-sha224@ssh.com
hmac-sha256-2@ssh.com
hmac-sha256@ssh.com
hmac-sha224@ssh.com
hmac-sha256@ssh.com
hmac-sha512@ssh.com
```

```
<kexs>
  <kex name="hmac-sha1" />
</kexs>

<kex>
  This element selects a KEX name that the client requests for the key exchange method.
</kex>

The KEXs are tried in the order they are specified.

<kex>
  This element selects a KEX name that the client requests for the key exchange method.
</kex>

The supported KEX methods 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
```

\(^2\)crypticore-mac@ssh.com is not supported on Tectia Server for Linux on IBM System z.
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

The default KEX methods used by the Connection Broker are, in order:

ecdh-sha2-nistp521
ecdh-sha2-nistp384
ecdh-sha2-nistp256
diffie-hellman-group-exchange-sha256
diffie-hellman-group14-sha1
diffie-hellman-group14-sha256
diffie-hellman-group14-sha256@ssh.com
diffie-hellman-group16-sha512
diffie-hellman-group18-sha512

<kexs>
  <kex name="diffie-hellman-group14-sha256@ssh.com" />
  <kex name="ecdh-sha2-nistp256" />
</kexs>

hostkey-algorithms

This element defines the host key signature algorithms used for server authentication. The algorithms that will be used are those that are defined in both Tectia Server and Connection Broker configuration files. This way the use of only certain algorithms, such as SHA-2, can be enforced by the server. The hostkey-algorithms element can contain multiple hostkey-algorithm elements.

The hostkey algorithms are tried in the order they are specified. Exception: If a host key of a server already exists in the host key store of the client, its algorithm is preferred.

hostkey-algorithm

This element selects a host key signature algorithm name to be used in server authentication with host keys or certificates.

The supported host key signature algorithms are:

\begin{verbatim}
rsa-sha2-256
rrsa-sha2-512
ssh-dss
\end{verbatim}

\begin{verbatim}
ecdsa-sha2-nistp521
x509v3-sign-dss
x509v3-sign-dss-sha224@ssh.com
x509v3-sign-dss-sha256@ssh.com
x509v3-sign-dss-sha384@ssh.com
\end{verbatim}
The default host key signature algorithms used by the Connection Broker are, in order:

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

```xml
<hostkey-algorithms>
  <hostkey-algorithm name="ssh-dss-sha512@ssh.com" />
  <hostkey-algorithm name="ssh-rsa-sha224@ssh.com" />
</hostkey-algorithms>
```

**rekey**

This element specifies the number of transferred bytes after which the key exchange is done again. The value "0" turns rekey requests off. This does not prevent the server from requesting rekeys, however. The default is 1000000000 (1 GB).

```xml
<rekey bytes="1000000000" />```

**authentication-methods**

This element specifies the authentication methods that are requested by the client-side components. The authentication-methods element can contain one of each: auth-hostbased, auth-password, auth-
publickey, auth-gssapi, and auth-keyboard-interactive. Alternatively, you can specify multiple authentication-method elements. The order of these elements is free.

The authentication methods are tried in the order the auth-* or authentication-method elements are listed. This means that the least interactive methods should be placed first.

When several interactive authentication methods are defined as allowed, Tectia client tools for z/OS will alternate between the methods and offers each of them in turn to the server in case the previous method failed.

**authentication-method**

This element specifies an authentication method name. It is included for backwards compatibility. Use the auth-* elements instead.

**auth-hostbased**

This element specifies that host-based authentication will be used.

The **auth-hostbased** element can include a local-hostname element.

**local-hostname**

This element specifies the local host name, as the value of the name attribute, that is advertised to the remote server during host-based authentication.

The remote server can use the client host name as a hint when locating the public key for the client host. This information is not significant to the authentication result, but makes it faster to find the relevant client host key, if the server has such a big storage of host identities, that trying them all would be infeasible.

**auth-password**

This element specifies that password authentication will be used.

**auth-publickey**

This element specifies that public-key authentication will be used.

The **auth-publickey** element can include a key-selection element.

The **auth-publickey** element can include a signature-algorithms attribute. The attribute defines the public-key signature algorithms used for client authentication, given as a comma-separated list. The algorithms that will be used are those that are defined in both Tectia Server and Connection Broker configuration files. This way the use of only certain algorithms, such as SHA-2, can be enforced. For a list of the supported algorithms, see hostkey-algorithm.
<authentication-methods>
  <auth-publickey signature-algorithms="ssh-dss,ssh-dss-sha512@ssh.com"/>
</authentication-methods>

key-selection

This element specifies the key selection policy the client uses when proposing user public keys to the server. The policy attribute can take the values automatic (default) and interactive-shy.

In the automatic mode, the client tries keys in the following order:

1. Keys with public key available and private key without a passphrase (no user interaction)
2. Keys with public key available but private key behind a passphrase (one passphrase query)
3. Keys that need a passphrase to get the public key but private key without passphrase (one user query for each key which is considered and proposed to server, but no user interaction for actual public-key login)
4. The rest of the keys, that is, keys that need a passphrase to get the public key and also to get the private key

In the interactive-shy mode, the client does not try any keys automatically, but it prompts the user to select the key from a list of available keys. If the authentication with the selected key fails, the client will prompt the user again, removing the already tried key(s) from the list. If there is only one key candidate available, the key will be tried automatically without asking the user.

The key-selection element can include the public-key and issuer-name elements.

public-key

This element can be used to specify that only plain public keys or only certificates are tried during public-key authentication. The type attribute can take the values plain and certificate. The default is to try both plain public keys and certificates.

issuer-name

This element can be used to filter the user certificates that will be included in the list presented to the user. The client-side user certificates can be filtered according to the issuer name that is compared to the certificate issuers requested or accepted by the server. The match-server-certificate attribute takes values yes and no. With value yes, Connection Broker tries matching the user certificate issuer name to the server certificate issuer name. Option no means that the issuer names are not used as a filter. By default, the filtering is not done.
The issuer-name is useful when a user has several certificates with different access rights to the same server, for example for a testing role and for an administrator role. The Connection Broker chooses the relevant certificates that are applicable on the remote host, and the user can choose the correct certificate from the short-listed ones.

**auth-keyboard-interactive**

This element specifies that keyboard-interactive methods will be used in authentication.

**auth-gssapi**

This element specifies that GSSAPI will be used in authentication.

**Note**

GSSAPI authentication is not available on z/OS.

The **auth-gssapi** element can take the following attributes:

The **dll-path** attribute specifies where the necessary GSSAPI libraries are located. If this attribute is not specified, the libraries are searched for in a number of common locations. The full path to the libraries should be given, for example, "/usr/lib/libkrb5.so,/usr/lib/libgssapi_krb5.so".

On AIX, the **dll-path** should include the archive file, if applicable, for example, ":/path/libgsapi_krb5.a(libgssapi_krb5.a.so)". The **archive(shared_object)** syntax is not necessary if the library is a shared object or has been extracted from the shared object.

The **allow-ticket-forwarding** attribute defines whether Tectia client tools for z/OS allows forwarding the Kerberos ticket over several connections. The attribute can have a value of yes or no. The default is no.

An example of authentication-methods configuration is shown below:

```xml
<authentication-methods>
  <auth-hostbased>
    <local-hostname name="host.example.com" />
  </auth-hostbased>
  <auth-gssapi allow-ticket-forwarding="yes"/>
  <auth-publickey>
    <key-selection policy="interactive-shy">
      <public-key type="plain" />
    </key-selection>
  </auth-publickey>
  <auth-keyboard-interactive />
  <auth-password>
    <password file="/path/filename" />
  </auth-password>
</authentication-methods>
```
hostbased-default-domain

This element specifies the host's default domain name (as name). This element is used to make sure the fully qualified domain name (FQDN) of the client host is transmitted to the server when using host-based user authentication.

Note

This element is not used by Tectia Server 6.6 for IBM z/OS. Instead the `DefaultDomain` keyword of `ssh2_config` is used for the same purpose. See *Tectia Server for IBM z/OS Administrator Manual* for more information.

compression

This element specifies whether the client sends the data compressed (PUT operation). When activated, compression is applied on-the-fly to all data sent out through the connection and on all channels in it.

The name of the compression algorithm and the compression level can be given as attributes. The `name` attribute can be defined as `none` (compression not used) or `zlib`, which utilizes z/OS zEDC zlib provided by IBM, and is currently the only supported algorithm. By default, compression is not used.

For zlib compression, the `level` attribute can be given an integer from 0 to 9. The default compression level is 6, when compression is activated but no level is given (or level is set to 0).

Example: to activate maximum level compression of sent data, make the following setting:

```xml
<compression name="zlib" level="9" />
```

Compression can also be activated per connection with command line tools. For information, see the `sshg3(1)`, `sftpg3(1)` and `scpg3(1)` man pages.

Note that this `compression` setting does not affect received data (GET operations), but their compression is defined on the Secure Shell server. Tectia Server always uses compression level 6.

proxy

This element defines rules for HTTP proxy or SOCKS servers the client will use for connections. It has a single attribute: `ruleset`.

The format of the attribute value is a sequence of rules delimited by semicolons (;). Each rule has a format that resembles the URL format. In a rule, the connection type is given first. The type can be `direct`, `socks`, `socks4`, `socks5`, or `http-connect` (socks is a synonym for socks4). This is followed by the server address and port. If the port is not given, the default ports are used: 1080 for SOCKS and 80 for HTTP.

After the address, zero or more conditions delimited by commas (,) are given. The conditions can specify IP addresses or DNS names.
direct://[cond[,cond]...];
socks://server/[cond[,cond]...];
socks4://server/[cond[,cond]...];
socks5://server/[cond[,cond]...];
http-connect://server/[cond[,cond]...]

The IP address/port conditions have an address pattern and an optional port range:

\[ip\_pattern[\:\[port\_range\]]\]

The \(ip\_pattern\) may have one of the following forms:

- a single IP address \(x.x.x.x\)
- an IP address range of the form \(x.x.x.x-y.y.y.y\)
- an IP sub-network mask of the form \(x.x.x.x/y\)

The DNS name conditions consist of a host name which may be a regular expression containing the characters "*" and "?" and a port range:

\[name\_pattern[\:\[port\_range\]]\]

An example proxy element is shown below. It causes the server to access the loopback address and the ssh.com domain directly, access *.*example with HTTP CONNECT, and all other destinations with SOCKS4.

<proxy ruleset="direct:///127.0.0.0/8,*.ssh.com;
  http-connect://http-proxy.ssh.com:8080/*.example;
  socks://fw.ssh.com:1080/" />

\textit{idle-timeout}

This element specifies how long idle time (after all connection channels are closed) is allowed for a connection before automatically closing the connection. The time is given in seconds. The type is always connection.

The default setting is 5 seconds. Setting a longer time allows the connection to the server to remain open even after a session (for example, sshg3) is closed. During this time, a new session to the server can be initiated without re-authentication. Setting the time to 0 (zero) terminates the connection immediately when the last channel to the server is closed.

<idle-timeout time="5" />

\textit{tcp-connect-timeout}

This element specifies a timeout for the TCP connection. When this setting is made, connection attempts to a Secure Shell server are stopped after the defined time if the remote host is down or unreachable. This timeout overrides the default system TCP timeout, and this timeout setting can be overridden by defining
a `tcp-connect-timeout` setting per connection profile (in the profiles settings) or per connection (on command line).

The time is given in seconds. The factory default value is 0. Value 0 (zero) disables this feature and the default system TCP timeout will be used.

```
<tcp-connect-timeout time="0" />
```

**keepalive-interval**

This element specifies an interval for sending keepalive messages to the Secure Shell server. The time value is given in seconds. The default setting is 0, meaning that the keepalive messages are disabled.

```
<keepalive-interval time="0" />
```

**exclusive-connection**

In the Connection Broker configuration file, the `exclusive-connection` element can be used to specify that a new connection is opened for each new channel. This setting takes one attribute `enable`, with value `yes` or `no`. The default is `no`, meaning that open connections are reused for new channels requested by a client.

On z/OS, the SOCKS Proxy always uses the exclusive connection type, meaning it opens a new connection for each new channel. This element is not needed in the SOCKS Proxy configuration file, and irrespective of the value set by the user, the SOCKS Proxy always interprets this setting as `exclusive-connection enable="yes"`.

**server-banners**

This element defines whether the server banner message file (if it exists) is visible to the user before login. The word `yes` or `no` is given as the value of the `visible` attribute. The default is `yes`.

To eliminate server banners:

```
<server-banners visible="no" />
```

**forwards**

This element contains `forward` elements that define whether X11 or agent forwarding (tunneling) are allowed on the client side.

- **Note**

  X11 forwarding is not available on z/OS.

**forward**

This element defines X11 or agent forwarding settings.
The `type` attribute defines the forwarding type (either `x11` or `agent`). The `state` attribute sets the forwarding `on`, `off`, or `denied`. If the forwarding is set as `denied`, the user cannot enable it on the command-line.

An example forward configuration, which denies X11 forwarding and allows agent forwarding globally, is shown below:

```xml
<forwards>
  <forward type="x11" state="denied" />
  <forward type="agent" state="on" />
</forwards>
```

**extended**

This element is reserved for future use.

**remote-environment**

This element contains `environment` elements which define the environment variables to be passed to the server from the client side. The environment variables are then set on the server when requesting a command, shell or subsystem.

Note that the server can restrict the setting of environment variables.

**environment**

This element defines the name and value of the environment variables, and whether the Connection Broker should process the value. Possible attributes are `name`, `value`, and `format`.

An example remote environment configuration:

```xml
<remote-environment>
  <environment name="FOO" value="bar" />
  <environment name="QUX" value="%Ubaz" format="yes" />
  <environment name="ZAPPA" value="%Ubaz" />
</remote-environment>
```

You can use `%U` in the `value` to indicate a user name. When `format="yes"` is also defined, the Connection Broker processes the `%U` into the actual user name before sending it to the server.

Let's assume the user name is `joedoe` in this example. The example configuration results in the following environment variables on the server side, provided that the server allows setting the environment variables:

- FOO=bar
- QUX=joedoebaz
- ZAPPA=%Ubaz

You can override the remote environment settings made in the configuration file if you use the `sshg3` command with the following arguments on the command-line client: `--remote-environment` or `--remote-environment-format`
server-authentication-methods

This `server-authentication-methods` element can be used to force the Connection Broker to use only certain methods in server authentication. This element can contain `auth-server-publickey` and `auth-server-certificate` elements (one of each). The order of these elements is free.

If only `auth-server-certificate` is specified, server certificate is needed. If no server certificate is received, connection fails.

If only `auth-server-publickey` is specified, (plain) server public key is needed. If no server public key is received, connection fails.

If both `auth-server-certificate` and `auth-server-publickey` are specified, server certificate is used if present. Otherwise server public key is used.

auth-server-certificate

The `auth-server-certificate` element specifies that certificates are used for server authentication.

auth-server-publickey

The `auth-server-publickey` element specifies that public host keys are used for server authentication.

Note

The host key policy settings have changed in version 6.1.4 and are now defined in the `auth-server-publickey` element.

The element takes attribute `policy` that defines how unknown server host keys are handled. It can have the following values:

- `strict`: Connect to the server only if the host key is found from the host key store and matches.

  If the policy is set to `strict`, the Connection Broker never adds host keys to the user's `.ssh2/hostkeys` directory upon connection, and refuses to connect to hosts whose key has changed. This provides maximum protection against man-in-the-middle attacks. However, it also means you must always obtain host keys via out-of-band means, which can be troublesome if you frequently connect to new hosts.

- `ask` (default): If the server host key is not found from the host key store, the user will be asked if he wants to accept the host key. If the host key has changed, the user is warned about it and asked how to proceed. If the client application is not able to ask the user (for example, `sftpg3` in batch mode, `-b`), the connection will be disconnected.
• **trust-on-first-use** or **tofu**: If the server host key is not found, it is stored to the user's .ssh2/hostkeys directory. If the host key has changed, the connection will be disconnected.

• **advisory**: Use of this setting effectively disables server authentication, which makes the connection vulnerable to active attackers.

If the server host key is not found in the host key store, it will be added to the user's .ssh2/hostkeys directory without user interaction. If the host key has changed, the connection will be continued without user interaction. The incident will be audited if logging is enabled.

When the policy is set to **advisory**, the keys from new hosts are automatically accepted and stored to the host key database without prompting acceptance from the user. However, changed host keys (from hosts whose keys are already in the database) are not stored, but they are accepted for that connection only.

This setting should be used only if logging is enabled for the Connection Broker.

⚠️ **Caution**

Consider carefully before setting the policy to **advisory**. Disabling the host-key checks makes you vulnerable to man-in-the-middle attacks.

In policy modes other than **strict**, if logging is enabled for the Connection Broker, Tectia client tools for z/OS will log information about changed and new host public keys with their fingerprints in the syslog (on Unix) or Event Viewer (on Windows).

**Note**

When transparent FTP tunneling or FTP-SFTP conversion is used, accepting the host key cannot be prompted from the user. Either this setting must be set to **tofu** in the ssh-socks-proxy-config.xml file or the host keys of the Secure Shell tunneling and SFTP servers must be obtained beforehand and stored based on the IP addresses of the servers, for example, by using the ssh-keydist-g3 key distribution tool. See Section 4.9.1.

If the **policy** attribute is not defined, the host key policy is interpreted based on the values of the old **strict-host-key-checking**, **host-key-always-ask**, and **accept-unknown-host-keys** options.

**Note**

In version 6.1.4 and later, the host key policy setting in the user-specific configuration file always takes precedence over the setting in the global configuration file.
authentication-method

The server-authentication-methods/authentication-method element specifies an authentication method name. This element is included for backwards compatibility. Use the auth-server-* elements instead.

```
<server-authentication-methods>
  <authentication-method name="publickey" />
  <authentication-method name="certificate" />
</server-authentication-methods>
```

An example server-authentication-methods element is shown below:

```
<server-authentication-methods>
  <auth-server-publickey policy="ask" />
  <auth-server-certificate />
</server-authentication-methods>
```

authentication-success-message

This setting defines whether the AuthenticationSuccessMsg messages are output. The authentication-success-message element takes attribute enable with value yes or no. The default is yes, meaning that the messages are output and logged.

sftpg3-mode

This setting defines how the sftp3 client behaves when transferring files. The sftpg3-mode element takes attribute compatibility-mode with the following values:

- **tectia** (the default) - sftpg3 transfers files recursively, meaning that files from the current directory and all its subdirectories are transferred.

- **ftp** - the get/put commands are executed as sget/sput meaning that they transfer a single file; and commands mget/mput have recursion depth set to 1 meaning that they only transfer files from the specified directory, not from subdirectories.

- **openssh** - commands get/put/mget/mput behave alike, and the recursion depth is set to 1, meaning that only files from the specified directory are transferred, not from subdirectories.

The recursion depth can be overridden by using the sftp3 client's commands get/put/mget/mput with command-line option --max-depth="LEVEL". For more information, see sftp3(1).

terminal-selection

This element defines how the Tectia terminal behaves when the user selects text with double-clicks. The element takes one attribute: selection-type, whose value can be:

- **select-words** - double-clicking selects one word at a time, space and all punctuation characters are used as delimiters. This is the default.
select-paths - selects strings of characters between spaces, meaning a selection is extended over characters \\/.-_, so that for example a path to a file can be selected by double-clicking anywhere in the path.

terminal-bell

This element defines whether Tectia terminal repeats audible notifications from the destination server. This option is only applied to connections with Unix servers. The element takes one attribute, bell-style, whose value can be:

none - no audible notifications are used

pc-speaker - the user’s PC speakers beep when an audible notification is indicated by the destination server

system-default - the Tectia terminal sounds the default alerts defined in the system on the destination server. This is the default.

close-window-on-disconnect

This element defines that also the Tectia terminal window is to be closed while disconnecting from a server session by pressing CTRL+D. The element takes one attribute, enable, whose value can be yes or no. The default is no meaning that CTRL+D closes only the server connection but the Tectia terminal window remains open.

quiet-mode

This setting defines whether the command line clients should suppress warnings, error messages and authentication success messages. The quiet-mode element takes attribute enable with value yes or no. The default is no, meaning that the errors and messages are output and logged.

The quiet-mode element affects command line tools scp3, ssh3, and sftpg3. Enabling the quiet mode here with setting quiet-mode enable="yes" is the same as running these clients with option -q. Note that the -q command line parameter will take priority over the quiet-mode element set in this configuration file.

checksum

The checksum element can be used to define a default setting for comparing checksums. This default overwrites the factory setting that checksums are not checked for files smaller than 32kB.

The checksum element takes attribute type, whose value can be:

yes|YES - MD5 checksums are checked on files larger than 32kB. This is the default value.

no|NO - checksums are not used.
md5|MD5 - only MD5 checksums are checked on files larger than 32kB. When the --fips parameter is set with the command line clients scp3 and sftp3, this hash is not used.

sha1|SHA1 - only SHA1 checksums are checked on files larger than 32kB. When the --fips parameter is set with the command line clients scp3 and sftp3, this hash is used.

md5-force|MD5-FORCE - MD5 checksums are forced, except when the --fips parameter is set with the command line tools scp3 and sftp3.

sha1-force|SHA1-FORCE - SHA1 checksums are forced on all files.

checkpoint|CHECKPOINT - checkpointing is forced on large files that are transferred one by one.

Note

If the Connection Broker is started in FIPS mode and the md5 attribute is defined in the configuration file, but scp3 or sftp3 are not started with the --fips parameter, then md5 is used.

Note that checksums can also be defined with the command line clients scp3 and sftp3, or with environment variables. The order of priority of the three checksum settings (in case they are different) is as follows, the later one always overwrites the previous value:

- checksum setting in the configuration file
- SSH_SFTP_CHECKSUM_MODE environment variable
- Command line arguments

address-family

The address-family element defines the IP address family. Give the address family as the value of the type attribute. Tectia client tools for z/OS will operate using IPv4 (inet) addressing, IPv6 (inet6), or both (any). The default value for type is any.

The profiles Element

The profiles element defines the connection profiles for connecting to the specified servers. Element profiles can contain multiple profile elements. Each profile defines the connection rules to one server. The settings in the profile element override the default connection settings.

When a profile is used for the connection, the settings in the profile override the default settings. See the section called “The default-settings Element”.

profile

The profile element defines a connection profile. It has the following attributes: id, name, host, port, protocol, host-type, connect-on-startup, user, and gateway-profile.
The profile id must be a unique identifier that does not change during the lifetime of the profile.

An additional name can be given to the profile. This is a free-form text string. The name can be used for connecting with the profile on the command line, so define a unique name for each profile.

The host attribute defines the address of the Secure Shell server host and it is a mandatory setting. The address can be either an IP address or a domain name. The value host="*" can be used to prompt the user to enter the host address when starting the session.

An empty value host="" can be used when the profile is used with transparent TCP or FTP tunneling or FTP-SFTP conversion and the host name is taken from the application (filter-engine/rule[@host-name-from-app="yes"]). See rule for details.

The port is a mandatory setting. It defines the port number of the Secure Shell server listener. The default port is 22.

The protocol is a mandatory setting. It defines the used communications protocol. Currently the only allowed value is secsh2.

If you want to make the connection specified by the profile automatically when the Connection Broker is started, set the value of the connect-on-startup attribute to yes. In this case, give also the user attribute (the user name the connection is made with). You also need to set up some form of non-interactive authentication for the connection.

The host-type attribute sets the server type for ASCII (text) file transfer. This specifies the line break convention that is used for ASCII files. The default value is default, meaning that the line break convention is determined by the local platform. If the client is running on Windows, Windows compatible line breaks (CR + LF, '\r\n') are used. If the client is running on any other platform, Unix compatible line breaks (LF, '\n') are used. Other possible values for host-type are windows (for Windows remote host) and unix (for Unix remote host). Define the value if you are using any other server than Tectia Server.

For FTP-SFTP conversion, set the server type here according to your target server to transfer ASCII files with correct line break convention.

The user attribute specifies the user name for opening the connection. The value "%USERNAME%" can be used to apply the user name of the currently logged in user. The value user="*" can be used to prompt the user to enter the user name when logging in. When the user attribute is not defined, the user name defined in the default connection settings will be used.

An empty value user="" can be used when the profile is used with transparent FTP tunneling or FTP-SFTP conversion and the user name is taken from the application (filter-engine/rule[@username-from-app="yes"]). See rule for details.

The gateway-profile attribute can be used to create nested tunnels. The tunnels defined under the local-tunnel element of the profile, and the tunnels defined under filter-engine and static-tunnels...
that refer to the profile can be nested. The profile name through which the connection is made is given as the value of the attribute. The first tunnel is created using the gateway host profile and from there the second tunnel is created to the host defined in this profile.

**hostkey**

This element gives the path to the remote server host public key file as a value of the file attribute.

Alternatively, the public key can be included as a base64-encoded ASCII block.

**ciphers**

This element defines the ciphers used with this profile. See ciphers for details.

**macs**

This element defines the MACs used with this profile. See macs for details.

**kexs**

This element defines the KEXs used with this profile. See kexs for details.

**hostkey-algorithms**

This element defines the hostkey signature algorithms used with this profile.

**rekey**

This element defines the rekeying settings used with this profile. See rekey for details.

**authentication-methods**

This element defines the authentication methods used with this profile. See authentication-methods for details.

**user-identities**

This element specifies the identities used in user public-key authentication. In contrast to the key-stores element that specifies all the keys that are available for the Connection Broker, this element can be used to control the keys that are attempted in authentication when this connection profile is used and to specify the order in which they are attempted.

The user-identities element can contain multiple identity elements. When multiple identity elements are used, they are tried out in the order they are listed.

**identity**

The identity element has the following attributes: identity-file, file, hash, id, and data.
The `identity-file` attribute specifies that the user identity is read in the identification file used with public-key authentication. Enter the full path to the file if it is located somewhere else than the default identification file directory which is `$HOME/.ssh2`. See also `ssh-broker-g3(1)`.

The `file` attribute specifies the path to the public-key file (primarily) or to a certificate. Enter the full path and file name as the value.

The `hash` attribute is used to enter the hash of the public key that will be used to identify the related private key. The key must be available for the Connection Broker. The public key hashes of the available keys can be listed with the `ssh-broker-ctl` tool. See also `ssh-broker-ctl(1)`.

The `id` attribute is reserved for future use.

The `data` attribute is reserved for future use.

An example `user-identities` element is shown below:

```xml
<user-identities>
  <identity identity-file="C:\\ mykey" />
  <identity file="$HOME/user/.ssh2/id_rsa_2048_a" />
  <identity file="C:\\private_keys\id_rsa_2048_a" />
  <identity hash="#a8edd3845005931aaa658b5573609e7d31e23af" />
</user-identities>
```

**compression**

This element defines the compression settings used with this profile. See `compression` for details.

**proxy**

This element defines the HTTP proxy and SOCKS server settings used with this profile. See `proxy` for details.

If `gateway-profile` has been defined for this profile, the proxy setting is ignored and the default proxy setting or the proxy setting of the gateway profile is used instead.

**idle-timeout**

This element defines the idle timeout settings used with this profile. See `idle-timeout` for details.

**tcp-connect-timeout**

This element defines the TCP connection timeout for this profile. The timeout is used to terminate connection attempts to Secure Shell servers that are down or unreachable. The default value is 0. See `tcp-connect-timeout` for details.
keepalive-interval

This element defines an interval for sending keepalive messages to the Secure Shell server. The setting applies to this profile. The default value is 0, meaning that no keepalive messages are sent. See keepalive-interval for details.

exclusive-connection

In the Connection Broker configuration, this element defines whether a new connection is opened for each new channel when a connection is made with this profile. This setting takes one attribute enable, with value yes or no. The default is no, meaning that open connections are reused for new channels requested by a client. See also exclusive-connection.

On z/OS, the SOCKS Proxy always uses the exclusive connection type, meaning it opens a new connection for each new channel. This element is not needed in the SOCKS Proxy configuration file, and irrespective of the value set by the user, the SOCKS Proxy always interprets this setting as: exclusive-connection enable="yes".

server-banners

This element defines the server banner setting used with this profile. See server-banners for details.

forwards

This element defines the forwards allowed with this profile. See forwards for details.

tunnels

The tunnels element defines the tunnels that are opened when a connection with this profile is made. The element can contain multiple local-tunnel and remote-tunnel elements.

local-tunnel

This element defines a local tunnel (port forwarding) that is opened automatically when a connection is made with the connection profile. It has five attributes: type, listen-port, listen-address, dst-host, dst-port, and allow-relay.

The type attribute defines the type of the tunnel. This can be tcp (default, no special processing), ftp (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session), or socks (Tectia client tools for z/OS will act as a SOCKS server for other applications, creating forwards as requested by the SOCKS transaction).

The listen-port attribute defines the listener port number on the local client.

The listen-address attribute can be used to define which network interfaces on the client should be listened. Its value can be an IP address belonging to an interface on the local host. Value 0.0.0.0 listens to all interfaces. The default is 127.0.0.1 (localhost loopback address on the client). Setting any other value requires setting allow-relay="yes".
For `address-family` option `inet6`, the default listen address is `::1`. To listen on all interfaces, specify `::`. For `address-family` option `any`, the listen address is both `127.0.0.1` and `::1` by default; to listen on all interfaces, specify `::`

Whenever a connection is made to the specified listener, the connection is tunneled over Secure Shell to the remote server and another connection is made from the server to a specified destination host and port (`dst-host`, `dst-port`). The connection from the server onwards will not be secure, it is a normal TCP connection.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = server host).

The `allow-relay` attribute defines whether connections to the listened port are allowed from outside the client host. The default is `no`. If you use `allow-relay="yes"`, it will check also the `listen-address` setting.

**remote-tunnel**

This element defines a remote tunnel (port forwarding) that is opened automatically when a connection is made with the connection profile. It has four attributes: `type`, `listen-port`, `listen-address`, `dst-host`, `dst-port`, and `allow-relay`.

The `type` attribute defines the type of the tunnel. This can be either `tcp` (default, no special processing) or `ftp` (temporary forwarding is created for FTP data channels, effectively securing the FTP session between the client and server).

The `listen-port` attribute defines the listener port number on the remote server.

The `listen-address` attribute can be used to define which network interfaces on the server should be listened. Its value can be an IP address belonging to an interface on the server host. Value `0.0.0.0` listens to all interfaces. The default is `127.0.0.1` (localhost loopback address on the server). Setting any other value requires that `allow-relay="yes"`.

For `address-family` option `inet6`, the default listen address is `::1`. To listen on all interfaces, specify `::`. For `address-family` option `any`, the listen address is both `127.0.0.1` and `::1` by default; to listen on all interfaces, specify `::`

Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified destination host and port (`dst-host`, `dst-port`). The connection from the client onwards will not be secure, it is a normal TCP connection.

The `dst-host` and `dst-port` attributes define the destination host address and port. The value of `dst-host` can be either an IP address or a domain name. The default is `127.0.0.1` (localhost = client host).
The `allow-relay` attribute defines whether connections to the listener port are allowed from outside the server host. The default is `no`.

extended
This element is reserved for future use.

remote-environment
This element defines the remote environment settings used with this profile. Within the `remote-environment` element, define an `environment` element for each environment variable to be passed to the server. See `remote-environment` for details.

server-authentication-methods
This element defines the server authentication methods allowed with this profile. See `server-authentication-methods` for details.

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

The password can be given directly in the `string` attribute, or a path to a file containing the password can be given in the `file` attribute, or a path to a program or a script that outputs the password can be given in the `command` attribute.

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

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

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

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

An example connection profile is shown below:

```xml
<profile name="rock"
   id="id1"
```
The **static-tunnels** Element

The **static-tunnels** setting is used to configure the behavior of the automatic tunnels. You can create listeners for local tunnels automatically when the Connection Broker starts up. The actual tunnel is formed the first time a connection is made to the listener port. If the connection to the server is not open at that time, it will be opened automatically as well.

The **static-tunnels** element can contain any number of **tunnel** elements.
tunnel

The tunnel element specifies a static tunnel. It has the following attributes: type, listen-port, listen-address, dst-host, dst-port, allow-relay, and profile.

The type attribute defines the type of the tunnel. This can be either tcp, ftp, or socks-proxy.

- tcp specifies a listener for generic TCP tunneling
- ftp specifies a listener for FTP tunneling (also the FTP data channels are tunneled)
- socks-proxy specifies a listener that acts as a SOCKS proxy towards the client applications. The traffic coming to the proxy is filtered using filter rules. When this option is used, a filter-engine element must be defined. See filter-engine.

The listen-port attribute defines the listener port number on the local client.

The listen-address attribute can be used to define which network interfaces on the client should be listened. Its value can be an IP address belonging to an interface on the local host. Value 0.0.0.0 listens to all interfaces. The default is 127.0.0.1 (localhost loopback address on the client). Setting any other value requires that allow-relay="yes".

For address-family option inet6, the default listen address is ::1. To listen on all interfaces, specify ::. For address-family option any, the listen address is both 127.0.0.1 and ::1 by default; to listen on all interfaces, specify ::.

The dst-host and dst-port attributes define the destination host address and port. The value of dst-host can be either an IP address or a domain name. The default is 127.0.0.1 (localhost = server host).

The allow-relay attribute defines whether connections to the listened port are allowed from outside the client host. The default is no.

The profile attribute specifies the connection profile ID that is used for the tunnel.

```
<static-tunnels>
  <tunnel type="tcp"
       listen-address="127.0.0.1"
       listen-port="9000"
       dst-host="st.example.com"
       dst-port="9000"
       allow-relay="no"
       profile="id1" />
</static-tunnels>
```

The filter-engine Element

The filter-engine element defines the filter rules for FTP-SFTP conversion and transparent FTP tunneling. On z/OS, also the static-tunnels element with a socks-proxy tunnel needs to be defined for the transparent FTP security to work.
Note

The SOCKS Proxy reads the filter-engine element from the global configuration file (/opt/tectia/etc/ssh-socks-proxy-config.xml), if such a file is available. Only when the global configuration file does not contain the filter-engine element, this element is read from the user-specific configuration file ($HOME/.ssh2/ssh-socks-proxy-config.xml).

The top level element is filter-engine. It has three attributes: ip-generate-start, ip6-generate-start and ftp-filter-at-signs (used with Tectia ConnectSecure, only).

The ip-generate-start attribute defines the start address of the pseudo IPv4 address space. Pseudo IPs are generated by the Connection Broker when applications do the DNS query through the SSH connection capture component.

The ip6-generate-start attribute is similar to ip-generate-start, but it defines the start address of the pseudo IPv6 address space.

Note

Under the filter-engine element there can be any amount of elements network, dns, filter, or rule. The order of the elements is important, because the filter engine uses the elements in the order they were specified in the configuration file.

network

The network element specifies a "location" where Tectia client tools for z/OS is running. By using the network element, you can implement location-awareness for Tectia client tools for z/OS. It has five attributes: id, address, domain, ip-generate-start and ip6-generate-start.

The id attribute specifies a unique identifier for the network element. The address attribute specifies the address of the network. It can be missing or empty, in which case it is not used. The domain attribute contains the domain name of the computer. It can also be missing or empty, in which case it is not used. The ip-generate-start attribute defines the start address of the pseudo IPv4 space. If it is defined here, it overrides the ip-generate-start attribute of the filter-engine element. The ip6-generate-start attribute is similar to ip-generate-start, but it defines the start address of the pseudo IPv6 address space.

dns

Note

The dns element exists for backward-compatibility reasons. Currently the rule element is used for the same settings.

The dns element creates a DNS rule for the filter engine. It has six attributes: id, network-id, application, host, ip-address, and pseudo-ip. For their descriptions, see rule below.
filter

Note

The filter element exists for backward-compatibility reasons. Currently the rule element is used for the same settings.

The filter element specifies an action for a connection. It has the following attributes: dns-id, ports, action, profile-id, destination, destination-port, fallback-to-plain.

The dns-id attribute is a reference to a dns element.

For the descriptions of the other attributes, see rule below.

rule

The rule element specifies how a filtered connection will be handled. It has the following attributes: application, host, ip-address, pseudo-ip, ports, action, profile-id, destination, destination-port, username, hostname-from-app, username-from-app, fallback-to-plain, show-sftp-server-banner.

The application attribute can be used to specify one or more applications to which the rule is applied. This can be a regular expression using the egrep syntax. For information on the syntax, see Appendix C.

The host attribute specifies a target host name. It can be a regular expression using the egrep syntax.

The ip-address attribute specifies the target host IP address. It can be a regular expression using the egrep syntax. In this case the Connection Broker does the string matching with the assumption that the IP address is written in its canonical form. If both the host name and the IP address are defined, the host attribute takes precedence and the ip-address attribute is ignored.

The pseudo-ip setting has the following effects when the ip-address is left empty and the host matches:

- When pseudo-ip="yes", the Connection Broker assigns a pseudo IP address for the target host and Tectia Server resolves the real IP address. The pseudo IP addresses should be used when accessing an internal network from the outside, because name resolution for the machines in the internal network is not available from the outside.

- When pseudo-ip="no", a normal DNS query is made for the target host name. The default value is no.

The ports attribute can be a single port or a range. A range is specified with a hyphen between two integers (for example *21-25*).

The action attribute specifies the action to be done when a filter matches. Its value can be DIRECT, BLOCK, TUNNEL, FTP-TUNNEL, or FTP-PROXY.
• **DIRECT** causes the connection to be made directly as plaintext without tunneling or FTP-SFTP conversion.

• **BLOCK** causes the connection to be blocked.

• **FTP-TUNNEL** activates transparent FTP tunneling

• **FTP-PROXY** causes the FTP-SFTP conversion to start and a connection to be made to the Secure Shell SFTP server.

The **profile-id** attribute can be used to specify the connection profile that defines the connection settings.

If the **profile-id** attribute is left empty and **hostname-from-app="yes"** is specified, the Secure Shell connection is made to the server specified by the client application using default settings. If a **profile-id** is specified and also **hostname-from-app="yes"** is specified, or the referred profile has * (an asterisk) or empty as the value of the **host** attribute, the Secure Shell connection is made to the server specified by the client application using the profile settings.

The **destination** and **destination-port** attributes can be used to define a static destination address and port number that will be used as the end point of the connection instead of the original address and port given by the application.

The **username** attribute can be used to define the user name used for connecting to the Secure Shell server, or you can define the path from where the Connection Broker should retrieve the user name.

The **hostname-from-app** attribute defines whether the Connection Broker should extract the Secure Shell server's host name from data sent by the application, or use a Secure Shell server defined by the connection profile in **profile-id**. The value is **yes** or **no**, and the default is **no**.

When **hostname-from-app="no"**, the tunnel or FTP-SFTP conversion will be created to the Secure Shell server specified in the connection profile referred in the **profile-id** attribute. Note that with transparent tunneling, the connection from the Secure Shell server to the final destination application will be unsecured and in plaintext. To achieve end-to-end security, the Secure Shell server should reside on the same host as the application.

When **hostname-from-app="yes"**, the tunnel or FTP-SFTP conversion will be created to the destination server specified by the application. This setting can be used with both FTP and TCP tunneling and FTP-SFTP conversion. When using **hostname-from-app="yes"**, it is no longer necessary to create a separate connection profile for each destination host. Note that this requires that a Secure Shell server is installed to each destination server (or, in the case of tunneling, that **fallback-to-plain** is enabled to allow direct connections to those servers that do not have Secure Shell installed).

The **username-from-app** attribute defines whether the FTP tunneling or FTP-SFTP conversion extracts the user name from data sent by the FTP application. The value is **yes** or **no**. The default is **no**.

When **username-from-app="yes"**, the user name received from the FTP client application is used. This setting can be used with FTP tunneling and FTP-SFTP conversion. This setting will override any user
name settings made in a related connection profile. When `username-from-app="no"`, the user name is
taken from the connection profile defined with the `profile-id` attribute.

In transparent tunneling, the `fallback-to-plain` attribute can be used to define whether a direct (unse-
cured) connection is used if creating the tunnel fails. The default value is `no`. Normally, when creating
the secure tunnel fails when applying a filter rule, the Connection Broker will return an error about not
being able to establish a connection.

Note
Do not enable the `fallback-to-plain` attribute for the FTP-SFTP conversion.

Note
Do not enable the `fallback-to-plain` and `pseudo-ip` options at the same time. If they both
are enabled, and a secure connection fails, the application will try a direct connection with the
pseudo IP, which will not work.

In FTP-SFTP conversion, if the target SFTP server is configured to send a banner to the client, the `show-
sftp-server-banner` attribute can be used to make the Connection Broker forward the SFTP server
banner to the FTP client. The allowed values for `show-sftp-server-banner` are `yes` and `no`. The default
value is `no` (the SFTP server banner is not forwarded to the FTP client).

Note
If you have made changes to the Connection Broker default configuration, make sure that
showing the server banner is enabled (server-banners visible="yes") in the default-
settings or in profiles.

Note
Sending the SFTP server banner to the FTP client will cause an extra connection opening to
the target SFTP server for retrieving the banner message.

In case of a failure in retrieving the banner message from the target SFTP server, the banner that Connection
Broker forwards to the FTP client includes an error description, a default banner, and the following text:

`Can't fetch banner from SFTP Server`

The `logging` Element
The `logging` element changes the logging settings that define the log event severities and logging facilities.
The element contains one or more `log-target` and `log-events` elements.

`log-target`
This element specifies the log target for auditing. By default, the broker does not log anything. This element
can be used to direct log data to a file or syslog.
The `log-target` element can have `file` and `type` as attributes.

The `type` attribute specifies the logging facility where the audit data is output to. The value can be `file`, `syslog` or `discard`.

The `file` attribute sets the file system path where the audit data is written to. If the `type` attribute has `syslog` or `discard` set, the `file` attribute is not allowed.

**log-events**

This element sets the severity and facility of different logging events. The events have reasonable default values, which are used if no explicit logging settings are made. This setting allows customizing the default values.

The element can also contain one or more `log-target` elements. When defined here, the `log-target` element will override the definition given in the `logging/log-target`.

For the events, `facility` and `severity` can be set as attributes. The events itself should be listed inside the `log-events` element.

The `facility` can be `normal`, `daemon`, `user`, `auth`, `local0`, `local1`, `local2`, `local3`, `local4`, `local5`, `local6`, `local7`, or `discard`. Setting the `facility` to `discard` causes the server to ignore the specified log events.

The `severity` can be `informational`, `notice`, `warning`, `error`, `critical`, `security-success`, or `security-failure`.

Any events that are not specifically defined in the configuration file will use the default values. The defaults can be overridden for all remaining events by giving an empty `log-events` element after all other definitions and by setting a severity value for it.

In the names of log events, the characters `'*'` and `'?'` can be used as wildcards.

An example logging configuration that logs all events, which are programmed to be logged by default, both to `/tmp/foo` and to `syslog`.

```xml
<logging>
  <log-target file="/tmp/foo" />
  <log-target type="syslog" />
</logging>
```

An example logging configuration in which events are logged to `/tmp/foo`, except those whose event name matches "`Key_store_*`", which will be discarded.

```xml
<logging>
  <log-target file="/tmp/foo" />
  <log-events facility="discard">
    Key_store_*
  </log-events>
</logging>
```
# A.2 Configuration File Quick Reference

This Appendix contains a quick reference to the elements of the XML-based configuration files `ssh-broker-config.xml` and `ssh-socks-proxy-config.xml`. The quick reference is divided into four tables:

- **Table A.1**: The general element
- **Table A.2**: The default-settings element
- **Table A.3**: The profiles element
- **Table A.4**: Other elements (static-tunnels, filter-engine, and logging)

The tables list the available configuration file elements with their attributes, attribute values (with the default value, if available, marked in bold) and descriptions. The element names in the tables are links that take you to detailed descriptions of the elements in `ssh-broker-config(5)`.

The element hierarchy is expressed with slashes (`/`) between parent and child elements.
<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes and their values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crypto-lib</td>
<td>mode = &quot;standard</td>
<td>fips&quot;</td>
</tr>
<tr>
<td>cert-validation</td>
<td>end-point-identity-check = &quot;yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>default-domain = domain_name</td>
<td>Default domain part of the remote system name</td>
</tr>
<tr>
<td></td>
<td>http-proxy-url = HTTP_proxy</td>
<td>HTTP proxy for making queries for certificate validity</td>
</tr>
<tr>
<td></td>
<td>socks-server-url = SOCKS_server</td>
<td>SOCKS server for making queries for certificate validity</td>
</tr>
<tr>
<td></td>
<td>max-path-length = number (default: &quot;10&quot;)</td>
<td>Maximum length of certification paths when validating certificates</td>
</tr>
<tr>
<td>cert-validation / ldap-server</td>
<td>address = LDAP_server_address</td>
<td>LDAP server address for fetching CRLs and/or subordinate CA certificates</td>
</tr>
<tr>
<td></td>
<td>port = port_number (default: &quot;389&quot;)</td>
<td>LDAP server port for fetching CRLs and/or subordinate CA certificates</td>
</tr>
<tr>
<td>cert-validation / ocsp-responder</td>
<td>url = URL_address</td>
<td>OCSP (Online Certificate Status Protocol) responder service address</td>
</tr>
<tr>
<td></td>
<td>validity-period = seconds (default: &quot;0&quot;)</td>
<td>Time period during which new OCSP queries for the same certificate are not made (the old result is used)</td>
</tr>
<tr>
<td>cert-validation / crl-prefetch</td>
<td>url = LDAP_URL</td>
<td>HTTP_URL</td>
</tr>
<tr>
<td></td>
<td>interval = seconds (default: &quot;3600&quot;)</td>
<td>How often the CRL is downloaded</td>
</tr>
<tr>
<td>cert-validation / dod-pki</td>
<td>enable = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td>cert-validation / ca-certificate</td>
<td>name = CA_name</td>
<td>Name of the certification authority (CA) used in server authentication</td>
</tr>
<tr>
<td></td>
<td>file = path</td>
<td>Path to the X.509 CA certificate file</td>
</tr>
<tr>
<td></td>
<td>disable-crls = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>use-expired-crls = seconds (default: &quot;0&quot;)</td>
<td>Time period for using expired CRLs</td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>cert-validation / key-store</strong> <em>(z/OS only)</em></td>
<td>type = &quot;zos-saf&quot;</td>
<td>CA certificates are stored in an external key store (System Authorization Facility) for server authentication</td>
</tr>
<tr>
<td></td>
<td>init = init_info</td>
<td>Key-store-provider-specific initialization info</td>
</tr>
<tr>
<td></td>
<td>disable-crls = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>use-expired-crls = seconds (default: &quot;0&quot;)</td>
<td>Time period for using expired CRLs</td>
</tr>
<tr>
<td><strong>key-stores / key-store</strong></td>
<td>type = &quot;mscapi</td>
<td>pkcs11</td>
</tr>
<tr>
<td></td>
<td>init = init_info</td>
<td>Key-store-provider-specific initialization info</td>
</tr>
<tr>
<td><strong>key-stores / user-keys</strong></td>
<td>directory = path</td>
<td>Directory where the user private keys are stored</td>
</tr>
<tr>
<td></td>
<td>passphrase-timeout = seconds (default: &quot;0&quot;)</td>
<td>Time after which the passphrase-protected private key will time out</td>
</tr>
<tr>
<td></td>
<td>passphrase-idle-timeout = seconds (default: &quot;0&quot;)</td>
<td>Time after which the passphrase-protected private key will time out unless the user accesses or uses the key</td>
</tr>
<tr>
<td><strong>key-stores / identification</strong></td>
<td>file = path</td>
<td>Location of the identification file that defines the user keys</td>
</tr>
<tr>
<td></td>
<td>base-path = path</td>
<td>Directory where the identification file expects the user private keys to be stored</td>
</tr>
<tr>
<td></td>
<td>passphrase-timeout = seconds (default: &quot;0&quot;)</td>
<td>Time after which the user must enter the passphrase again</td>
</tr>
<tr>
<td></td>
<td>passphrase-idle-timeout = seconds (default: &quot;0&quot;)</td>
<td>Time after which the passphrase times out if there are no user actions</td>
</tr>
<tr>
<td><strong>user-config-directory</strong></td>
<td>path = path (default: &quot;%USER_CONFIG_DIRECTORY%&quot;)</td>
<td>Non-default location of user-specific configuration files</td>
</tr>
<tr>
<td><strong>file-access-control</strong> <em>(Unix only)</em></td>
<td>enable = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td><strong>protocol-parameters</strong></td>
<td>threads = number (if set to 0, default value is used)</td>
<td>The number of threads the protocol library uses (fast path dispatcher threads)</td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>known-hosts</strong></td>
<td>path = path</td>
<td>Non-default location of known hosts file or directory</td>
</tr>
<tr>
<td></td>
<td>file = path</td>
<td>Location of OpenSSH-style known_hosts file</td>
</tr>
<tr>
<td></td>
<td>directory = path</td>
<td>Non-default directory for storing known host keys</td>
</tr>
<tr>
<td></td>
<td>filename-format = &quot;hash</td>
<td>plain</td>
</tr>
<tr>
<td><strong>known-hosts / key-store</strong> <em>(z/OS only)</em></td>
<td>type = &quot;zos-saf&quot;</td>
<td>Certificates of known server hosts are stored in an external key store</td>
</tr>
<tr>
<td></td>
<td>init = init_info</td>
<td>Key-store-provider-specific initialization info</td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>user = user_name</td>
<td>Default user name to be used when connecting to remote servers</td>
<td></td>
</tr>
<tr>
<td>name = cipher_name</td>
<td>A cipher that the client requests for data encryption</td>
<td></td>
</tr>
<tr>
<td>name = MAC_name</td>
<td>A MAC that the client requests for data integrity verification</td>
<td></td>
</tr>
<tr>
<td>name = KEX_name</td>
<td>A KEX that the client requests for the key exchange method</td>
<td></td>
</tr>
<tr>
<td>name = hostkey-algorithm_name</td>
<td>A host key signature algorithm to be used in server authentication with host keys or certificates</td>
<td></td>
</tr>
<tr>
<td>bytes = number (default: &quot;1000000000&quot; (1 GB))</td>
<td>Number of transferred bytes after which key exchange is done again</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Host-based authentication will be used</td>
<td></td>
</tr>
<tr>
<td>name = host_name</td>
<td>Local host name that is advertised to the remote server during host-based authentication</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Password authentication will be used</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Public-key authentication will be used</td>
<td></td>
</tr>
<tr>
<td>signature-algorithms = comma-separated_list</td>
<td>Public-key signature algorithms used for client authentication</td>
<td></td>
</tr>
<tr>
<td>policy - &quot;automatic</td>
<td>interactive-shy&quot;</td>
<td>Key selection policy used by the client when proposing user public keys to the server</td>
</tr>
<tr>
<td>type = &quot;plain</td>
<td>certificate&quot; (by default, both are tried)</td>
<td>Only plain public keys or only certificates are tried during public-key authentication</td>
</tr>
<tr>
<td>name = certificate_issuer_name</td>
<td>Client-side user certificates can be filtered by comparing this name to the certificate issuers requested or accepted by the server</td>
<td></td>
</tr>
<tr>
<td>match-server-certificate - &quot;yes</td>
<td>no&quot;</td>
<td>The Connection Broker tries matching the user certificate issuer name to the server certificate issuer name</td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>authentication-methods / auth-gssapi</strong></td>
<td></td>
<td>GSSAPI will be used in authentication</td>
</tr>
<tr>
<td></td>
<td>dll-path = path (ignored on Windows)</td>
<td>Location of the necessary GSSAPI libraries</td>
</tr>
<tr>
<td></td>
<td>allow-ticket-forwarding = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td><strong>authentication-methods / auth-keyboard-interactive</strong></td>
<td></td>
<td>Keyboard-interactive methods will be used in authentication</td>
</tr>
<tr>
<td><strong>hostbased-default-domain</strong></td>
<td>name = domain_name</td>
<td>Host's default domain name that is appended to the short host name before transmitting it to the server</td>
</tr>
<tr>
<td><strong>compression</strong></td>
<td>name = &quot;none</td>
<td>zlib&quot;</td>
</tr>
<tr>
<td></td>
<td>level = [0 to 9] (default: &quot;0&quot; (= level 6))</td>
<td>For zlib, compression level.</td>
</tr>
<tr>
<td><strong>proxy</strong></td>
<td>ruleset = rule_sequence</td>
<td>Rules for HTTP proxy or SOCKS servers the client will use for connections</td>
</tr>
<tr>
<td><strong>idle-timeout</strong></td>
<td>type = &quot;connection&quot;</td>
<td>Idle timeout is always defined for connections</td>
</tr>
<tr>
<td></td>
<td>time = seconds (default: &quot;5&quot;)</td>
<td>Idle time (after all connection channels are closed) allowed for a connection before automatically closing the connection</td>
</tr>
<tr>
<td><strong>tcp-connect-timeout</strong></td>
<td>time = seconds (default: &quot;0&quot;)</td>
<td>Timeout for TCP connections (after which connection attempts to a Secure Shell server are stopped if the remote host is down or unreachable)</td>
</tr>
<tr>
<td><strong>keepalive-interval</strong></td>
<td>time = seconds (default: &quot;0&quot;)</td>
<td>Time interval for sending keepalive messages to the Secure Shell server</td>
</tr>
<tr>
<td><strong>exclusive-connection</strong></td>
<td>enable = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td><strong>server-banners</strong></td>
<td>visible = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td><strong>forwards / forward</strong></td>
<td>type = &quot;x11</td>
<td>agent&quot;</td>
</tr>
<tr>
<td></td>
<td>state = &quot;on</td>
<td>off</td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>remote-environment / environment</td>
<td>name = env_var_name</td>
<td>Name of an environment variable that is to be passed to the server from the client side</td>
</tr>
<tr>
<td></td>
<td>value = string</td>
<td>Value of the environment variable</td>
</tr>
<tr>
<td></td>
<td>format = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td>server-authentication-methods / auth-server-certificate</td>
<td>-</td>
<td>Use certificates for server authentication</td>
</tr>
<tr>
<td>server-authentication-methods / auth-server-publickey</td>
<td>-</td>
<td>Use public host keys for server authentication</td>
</tr>
<tr>
<td></td>
<td>policy</td>
<td>Policy for handling unknown server host keys</td>
</tr>
<tr>
<td></td>
<td>enable = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td>authentication-success-message</td>
<td>compatibility-mode</td>
<td>Behavior of sftpg3 when transferring files</td>
</tr>
<tr>
<td></td>
<td>- &quot;tectia</td>
<td>ftp</td>
</tr>
<tr>
<td>terminal-selection</td>
<td>selection-type</td>
<td>Behavior of the Tectia terminal when the user selects text with double-clicks</td>
</tr>
<tr>
<td></td>
<td>- &quot;select-words</td>
<td>select-paths&quot;</td>
</tr>
<tr>
<td>terminal-bell</td>
<td>bell-style = &quot;none</td>
<td>pc-speaker</td>
</tr>
<tr>
<td>close-window-on-disconnect</td>
<td>enable = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td>quiet-mode</td>
<td>enable = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td>checksum</td>
<td>type = &quot;yes</td>
<td>no</td>
</tr>
<tr>
<td>address-family</td>
<td>type = &quot;any</td>
<td>inet</td>
</tr>
</tbody>
</table>
Table A.3. Configuration File Quick Reference - the `profiles` element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes and their values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile</td>
<td>id = id</td>
<td>Unique identifier that does not change during the lifetime of the profile</td>
</tr>
<tr>
<td></td>
<td>name = string</td>
<td>Unique name (free-form text string) that can be used for connecting with the profile on the command line</td>
</tr>
<tr>
<td></td>
<td>host = IP_address</td>
<td>FQDN</td>
</tr>
<tr>
<td></td>
<td>port = port_number</td>
<td>Secure Shell server listener port number</td>
</tr>
<tr>
<td></td>
<td>(default: &quot;22&quot;)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>protocol = &quot;secsh2&quot;</td>
<td>The communications protocol used by the profile</td>
</tr>
<tr>
<td></td>
<td>host-type = &quot;default</td>
<td>windows</td>
</tr>
<tr>
<td></td>
<td>connect-on-startup = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>user = user_name</td>
<td>User name for opening the connection</td>
</tr>
<tr>
<td></td>
<td>gateway-profile = profile_name</td>
<td>Create nested tunnels</td>
</tr>
<tr>
<td>profile / hostkey</td>
<td>file = path</td>
<td>Path to the remote server host public key file</td>
</tr>
<tr>
<td>profile / ciphers / cipher</td>
<td>name = cipher_name</td>
<td>A cipher used with this profile</td>
</tr>
<tr>
<td>profile / macs / mac</td>
<td>name = MAC_name</td>
<td>A MAC used with this profile</td>
</tr>
<tr>
<td>profile / kexs / kex</td>
<td>name = KEX_name</td>
<td>A KEX used with this profile</td>
</tr>
<tr>
<td>profile / hostkey-algorithms / hostkey-algorithm</td>
<td>name = hostkey-algorithm_name</td>
<td>Host key signature algorithm used with this profile</td>
</tr>
<tr>
<td>profile / rekey</td>
<td>bytes = number</td>
<td>Number of transferred bytes after which key exchange is done again when using this profile</td>
</tr>
<tr>
<td>(default: &quot;1000000000&quot; (1 GB))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>profile / authentication-methods</td>
<td>Define the authentication methods for this profile using the same child elements as with default-settings / authentication-methods (see Table A.2)</td>
<td></td>
</tr>
<tr>
<td>profile / user-identities / identity</td>
<td>identity-file = path</td>
<td>The user identity is read in the identification file used with public-key authentication</td>
</tr>
<tr>
<td>file = path</td>
<td>Path to the public-key file (primarily) or to a certificate</td>
<td></td>
</tr>
<tr>
<td>hash = hash</td>
<td>Hash of the public key that will be used to identify the related private key</td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>profile / compression</td>
<td>name = &quot;none</td>
<td>zlib&quot;</td>
</tr>
<tr>
<td></td>
<td>level = [0 to 9] (default: &quot;0&quot; (= level 6))</td>
<td>For zlib, compression level.</td>
</tr>
<tr>
<td>profile / proxy</td>
<td>ruleset = rule_sequence</td>
<td>Rules for HTTP proxy or SOCKS servers the client will use for connections with this profile</td>
</tr>
<tr>
<td>profile / idle-timeout</td>
<td>type = &quot;connection&quot;</td>
<td>Idle timeout is always defined for connections</td>
</tr>
<tr>
<td></td>
<td>time = seconds (default: &quot;5&quot;)</td>
<td>Idle time (after all connection channels are closed) allowed for a connection before automatically closing the connection opened with this profile</td>
</tr>
<tr>
<td>profile / tcp-connect-timeout</td>
<td>time = seconds (default: &quot;5&quot;)</td>
<td>Timeout for TCP connections with this profile: Connection attempts to a Secure Shell server are stopped after the defined time if the remote host is down or unreachable</td>
</tr>
<tr>
<td>profile / keepalive-interval</td>
<td>time = seconds (default: &quot;0&quot;)</td>
<td>Time interval for sending keepalive messages to the Secure Shell server with this profile</td>
</tr>
<tr>
<td>profile / exclusive-connection</td>
<td>enable = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td>profile / server-banners</td>
<td>visible = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td>profile / forwards / forward</td>
<td>type = &quot;x11</td>
<td>agent&quot;</td>
</tr>
<tr>
<td></td>
<td>state = &quot;on</td>
<td>off</td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>profile / tunnels / local-tunnel</strong></td>
<td>type = &quot;tcp</td>
<td>ftp</td>
</tr>
<tr>
<td></td>
<td>listen-address = IP_address (default: 127.0.0.1)</td>
<td>The network interfaces that should be listened on the client</td>
</tr>
<tr>
<td></td>
<td>listen-port = port_number</td>
<td>Listener port number on the local client</td>
</tr>
<tr>
<td></td>
<td>dst-host = IP_address</td>
<td>domain_name (default: 127.0.0.1)</td>
</tr>
<tr>
<td></td>
<td>dst-port = port_number</td>
<td>Destination port</td>
</tr>
<tr>
<td></td>
<td>allow-relay = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td><strong>profile / tunnels / remote-tunnel</strong></td>
<td>type = &quot;tcp</td>
<td>ftp&quot;</td>
</tr>
<tr>
<td></td>
<td>listen-address = IP_address (default: 127.0.0.1)</td>
<td>The network interfaces that should be listened on the server</td>
</tr>
<tr>
<td></td>
<td>listen-port = port_number</td>
<td>Listener port number on the remote server</td>
</tr>
<tr>
<td></td>
<td>dst-host = IP_address</td>
<td>domain_name (default: 127.0.0.1)</td>
</tr>
<tr>
<td></td>
<td>dst-port = port_number</td>
<td>Destination port</td>
</tr>
<tr>
<td></td>
<td>allow-relay = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td><strong>profile / remote-environment / environment</strong></td>
<td>name = env_var_name</td>
<td>Name of an environment variable that is to be passed to the server from the client side</td>
</tr>
<tr>
<td></td>
<td>value = string</td>
<td>Value of the environment variable</td>
</tr>
<tr>
<td></td>
<td>format = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td><strong>profile / server-authentication-methods</strong></td>
<td></td>
<td>Define the server authentication methods allowed with this profile using the same child elements as with default-settings / server-authentication-methods (see Table A.2)</td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>profile / password</td>
<td>string = password</td>
<td>User password that the client will send as a response to password authentication</td>
</tr>
<tr>
<td></td>
<td>file = password_file</td>
<td>File containing the password</td>
</tr>
<tr>
<td></td>
<td>command = path</td>
<td>Path to a program or script that outputs the password</td>
</tr>
</tbody>
</table>
Table A.4. Configuration File Quick Reference - the static-tunnels, filter-engine, and logging elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes and their values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static-tunnels</strong></td>
<td>type = &quot;tcp</td>
<td>ftp&quot;</td>
</tr>
<tr>
<td><strong>tunnel</strong></td>
<td>listen-address = IP_address (default: 127.0.0.1)</td>
<td>The network interfaces that should be listened on the client</td>
</tr>
<tr>
<td></td>
<td>listen-port = port_number</td>
<td>Listener port number on the local client</td>
</tr>
<tr>
<td></td>
<td>dst-host</td>
<td>Destination host address</td>
</tr>
<tr>
<td></td>
<td>- IP_address</td>
<td>domain_name (default: 127.0.0.1)</td>
</tr>
<tr>
<td></td>
<td>dst-port = port_number</td>
<td>Destination port</td>
</tr>
<tr>
<td></td>
<td>allow-relay = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>profile = ID</td>
<td>Connection profile ID that is used for the tunnel</td>
</tr>
<tr>
<td><strong>filter-engine</strong></td>
<td>ip-generate-start</td>
<td>Start address of the pseudo IPv4 address space</td>
</tr>
<tr>
<td></td>
<td>- IPv4_address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ip6-generate-start</td>
<td>Start address of the pseudo IPv6 address space</td>
</tr>
<tr>
<td></td>
<td>- IPv6_address</td>
<td></td>
</tr>
<tr>
<td><strong>filter-engine</strong></td>
<td>id = ID</td>
<td>Unique identifier for the element</td>
</tr>
<tr>
<td></td>
<td>address = network_address</td>
<td><em>(Optional)</em> network address</td>
</tr>
<tr>
<td></td>
<td>domain = domain_name</td>
<td>Domain name of the computer</td>
</tr>
<tr>
<td><strong>/ network</strong></td>
<td>ip-generate-start</td>
<td>Start address of the pseudo IPv4 address space</td>
</tr>
<tr>
<td></td>
<td>- IPv4_address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ip6-generate-start</td>
<td>Start address of the pseudo IPv6 address space</td>
</tr>
<tr>
<td></td>
<td>- IPv6_address</td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Attributes and their values</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>filter-engine / rule</td>
<td>application = application</td>
<td>One or more applications to which the rule is applied. Regular expressions (egrep) can be used.</td>
</tr>
<tr>
<td></td>
<td>host = host_name</td>
<td>Filtered connection’s target host name. Regular expressions (egrep) can be used.</td>
</tr>
<tr>
<td></td>
<td>ip-address = IP_address</td>
<td>Filtered connection’s target host IP address. Regular expressions (egrep) can be used.</td>
</tr>
<tr>
<td></td>
<td>pseudo-ip = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>ports = port_number</td>
<td>port_range</td>
</tr>
<tr>
<td></td>
<td>action = &quot;direct</td>
<td>block</td>
</tr>
<tr>
<td></td>
<td>profile-id = ID</td>
<td>The connection profile that defines the connection settings</td>
</tr>
<tr>
<td></td>
<td>destination = address</td>
<td>Static destination address that will be used as the end point of the connection</td>
</tr>
<tr>
<td></td>
<td>destination-port = port_number</td>
<td>Static destination port that will be used as the end point of the connection</td>
</tr>
<tr>
<td></td>
<td>username = user_name</td>
<td>path</td>
</tr>
<tr>
<td></td>
<td>hostname-from-app = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>username-from-app = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>fallback-to-plain = &quot;yes</td>
<td>no&quot;</td>
</tr>
<tr>
<td></td>
<td>show-sftp-server-banner = yes</td>
<td>no</td>
</tr>
</tbody>
</table>
### A.3 Broker Configuration File Syntax

The DTD of the Connection Broker configuration file is shown below:

```xml
<!-- secsh-broker.dtd -->
<!-- Copyright (c) 2017 SSH Communications Security Corporation. -->
<!-- This software is protected by international copyright laws. -->
<!-- All rights reserved. -->
<!-- Document type definition for the Connection Broker XML configuration files. -->
<!-- Tunable parameters used in the policy. -->
<!-- Both ipv4 and ipv6 are enabled by default -->
<!ENTITY default-address-family-type "any">
<!-- The top-level element -->
<!ATTLIST secsh-broker
  version CDATA #IMPLIED>
<!-- General element. Only "known-hosts" can appear multiple times. -->
```
<!ELEMENT general               (crypto-lib|cert-validation|key-stores|strict-host-key-checking|host-key-always-ask|accept-unknown-host-keys|known-hosts|user-config-directory|file-access-control|protocol-parameters)*>
<!-- Cryptographic library. -->
<!ELEMENT crypto-lib            EMPTY>
<!ATTLIST crypto-lib
mode                      (fips|standard) "standard">
<!-- PKI settings. "dod-pki" element may appear only once, other elements -->
<!-- may be specified multiple times. -->
<!ELEMENT cert-validation       (ldap-server|ocsp-responder|crl-prefetch|dod-pki|ca-certificate|key-store)*>
<!ATTLIST cert-validation
end-point-identity-check  (yes|no|YES|NO|ask|ASK) "yes"
default-domain            CDATA           #IMPLIED
http-proxy-url            CDATA           #IMPLIED
socks-server-url          CDATA           #IMPLIED
max-path-length           CDATA           "10">
<!ELEMENT ldap-server           EMPTY>
<!ATTLIST ldap-server
address                   CDATA           #REQUIRED
port                      CDATA           "389">
<!ELEMENT ocsp-responder        (#PCDATA)>  
<!ATTLIST ocsp-responder
url                       CDATA           #REQUIRED
validity-period           CDATA           "0"
responder-certificate     CDATA           #IMPLIED>
<!-- CRL prefetch. -->
<!ELEMENT crl-prefetch          EMPTY>
<!ATTLIST crl-prefetch
interval                  CDATA           "3600"
url                       CDATA           #REQUIRED>
<!-- CA certificates. -->
<!ELEMENT ca-certificate        (#PCDATA)>  
<!ATTLIST ca-certificate
name                      CDATA           #REQUIRED
file                      CDATA           #IMPLIED
disable-crls              (yes|no|YES|NO) "no"
use-expired-crls CDATA "0">

<!ELEMENT dod-pki EMPTY>
<!ATTLIST dod-pki enable (yes|no|YES|NO) "no">

<!ELEMENT key-stores ((key-store|user-keys|identification)*)>
<!ELEMENT key-store EMPTY>
<!ATTLIST key-store type CDATA #REQUIRED init CDATA #IMPLIED disable-crls (yes|no|YES|NO) "no" use-expired-crls CDATA "0">

<!ELEMENT user-keys EMPTY>
<!ATTLIST user-keys directory CDATA #IMPLIED poll-interval CDATA "10" passphrase-timeout CDATA "0" passphrase-idle-timeout CDATA "0">

<!ELEMENT identification EMPTY>
<!ATTLIST identification file CDATA #REQUIRED base-path CDATA #IMPLIED passphrase-timeout CDATA "0" passphrase-idle-timeout CDATA "0">

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT strict-host-key-checking EMPTY>
<!ATTLIST strict-host-key-checking enable (yes|no|YES|NO) #REQUIRED>

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT host-key-always-ask EMPTY>
<!ATTLIST host-key-always-ask enable (yes|no|YES|NO) #REQUIRED>

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT accept-unknown-host-keys EMPTY>
<!ATTLIST accept-unknown-host-keys enable (yes|no|YES|NO) #REQUIRED>

<!ELEMENT exclusive-connection EMPTY>
<!ATTLIST exclusive-connection enable (yes|no|YES|NO) #REQUIRED>

<!ELEMENT known-hosts (key-store*)>
<!ATTLIST known-hosts>
<!-- Extended plugin configuration -->
<!ELEMENT extended (ext)*>
<!ELEMENT ext (#PCDATA | EMPTY | ext)>
<!ATTLIST ext
name CDATA #REQUIRED>

<!-- Default settings element. No element may appear multiple times. -->
<!ELEMENT default-settings (ciphers|macs|kexs|hostkey-algorithms|
transport-distribution|rekey|
authentication-methods|
hostbased-default-domain|
compression|proxy|idle-timeout|
tcp-connect-timeout|keepalive-interval|
exclusive-connection|server-banners|
forwards|extended|remote-environment|
servers-authentication-methods|
authentication-success-message|
sftpg3-mode|terminal-selection|terminal-bell|
close-window-on-disconnect|quiet-mode|
checksum|address-family)>
<!ATTLIST default-settings
user CDATA #IMPLIED>

<!-- Server banners. -->
<!ELEMENT server-banners EMPTY>
<!ATTLIST server-banners
visible (yes|no|YES|NO) "yes">

<!-- Ciphers element. -->
<!ELEMENT ciphers (cipher*)>
<!ELEMENT cipher EMPTY>
<!ATTLIST cipher
name CDATA #REQUIRED>

<!-- Macs element. -->
<!ELEMENT macs (mac*)>
<!ELEMENT mac EMPTY>
<!ATTLIST mac
name CDATA #REQUIRED>
<!-- Kexs element. -->
<!ELEMENT kexs                  (kex*)>

<!-- Kex. -->
<!ELEMENT kex                   EMPTY>
<!ATTLIST kex
  name                      CDATA           #REQUIRED>

<!-- Hostkey algorithms element. -->
<!ELEMENT hostkey-algorithms (hostkey-algorithm*)>

<!-- Hostkey algorithm. -->
<!ELEMENT hostkey-algorithm    EMPTY>
<!ATTLIST hostkey-algorithm
  name                      CDATA           #REQUIRED>

<!ELEMENT rekey                 EMPTY>
<!ATTLIST rekey
  bytes                     CDATA           "0">

<!-- Hostbased default domain. -->
<!ELEMENT hostbased-default-domain EMPTY>
<!ATTLIST hostbased-default-domain
  name                      CDATA           #REQUIRED>

<!-- Authentication methods element. -->
<!ELEMENT authentication-methods (authentication-method|auth-hostbased
  |auth-password|auth-publickey|auth-gssapi
  |auth-keyboard-interactive|auth-server-publickey
  |auth-server-certificate)*>

<!ELEMENT server-authentication-methods (authentication-method
  |auth-server-publickey
  |auth-server-certificate)>

<!ELEMENT auth-server-publickey EMPTY>
<!ATTLIST auth-server-publickey
  policy                    CDATA #IMPLIED><!-- "strict", "ask", "tofu", -->

<!ELEMENT auth-server-certificate   EMPTY>

<!ELEMENT remote-environment    (environment*)>

<!ELEMENT environment           EMPTY>
<!ATTLIST environment
  name                      CDATA           #REQUIRED
  value                     CDATA           #REQUIRED
  format                    (yes|no|YES|NO) "no">

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT transport-distribution EMPTY>
<!ATTLIST transport-distribution>
num-transports CDATA #REQUIRED>

<!-- This element is deprecated and included for backwards compatibility only -->
<!ELEMENT authentication-method EMPTY>
<!ATTLIST authentication-method
  name CDATA #REQUIRED>

<!ELEMENT auth-hostbased (local-hostname?)>
<!ELEMENT local-hostname EMPTY>
<!ATTLIST local-hostname
  name CDATA #REQUIRED>

<!ELEMENT auth-password EMPTY>

<!ELEMENT auth-publickey (key-selection?)>
<!ATTLIST auth-publickey
  signature-algorithms CDATA #IMPLIED>

<!ELEMENT key-selection (public-key|issuer-name|subject-name|
  extended-key-usage|key-usage|policy-info)*>
<!ATTLIST key-selection
  policy CDATA #IMPLIED
  exclude (yes|no|YES|NO) "no"
  require-all (yes|no|YES|NO) "no">

<!ELEMENT public-key EMPTY>
<!ATTLIST public-key
  type CDATA #REQUIRED>

<!ELEMENT issuer-name EMPTY>
<!ATTLIST issuer-name
  name CDATA #IMPLIED
  pattern CDATA #IMPLIED
  match-server-certificate (yes|no|YES|NO) "no">

<!ELEMENT subject-name EMPTY>
<!ATTLIST subject-name
  name CDATA #IMPLIED
  pattern CDATA #IMPLIED>

<!ELEMENT extended-key-usage (#PCDATA)>
<!ATTLIST extended-key-usage
  oid CDATA #IMPLIED
  explicit (yes|no|YES|NO) "no">

<!ELEMENT key-usage (#PCDATA)>
<!ATTLIST key-usage
  bit CDATA #IMPLIED>

<!ELEMENT auth-keyboard-interactive EMPTY>
<!ELEMENT auth-gssapi EMPTY>

<!-- Actually, the default for allow-ticket-forwarding is "no", but we
don't want to override value if it is left undefined. -->
<!ATTLIST auth-gssapi
dll-path CDATA "/usr/lib/libgssapi_krb5.so,"
allow-ticket-forwarding (yes|no) #IMPLIED>

<!-- User identities. -->
<!ELEMENT user-identities (identity*)>
<!ELEMENT identity EMPTY>
<!ATTLIST identity
  identity-file CDATA #IMPLIED
  file CDATA #IMPLIED
  hash CDATA #IMPLIED
  id CDATA #IMPLIED
  data CDATA #IMPLIED>

<!-- Password. -->
<!ELEMENT password (#PCDATA)>
<!ATTLIST password
  string CDATA #IMPLIED
  file CDATA #IMPLIED
  command CDATA #IMPLIED>

<!-- Proxy rules. -->
<!ELEMENT proxy EMPTY>
<!ATTLIST proxy
  ruleset CDATA #REQUIRED>

<!-- Idle timeout. -->
<!ELEMENT idle-timeout EMPTY>
<!ATTLIST idle-timeout
  type (connection) "connection"
  time CDATA #IMPLIED>

<!-- Connect timeout. -->
<!ELEMENT tcp-connect-timeout EMPTY>
<!ATTLIST tcp-connect-timeout
  time CDATA #REQUIRED>

<!-- Keepalive interval. -->
<!ELEMENT keepalive-interval EMPTY>
<!ATTLIST keepalive-interval
  data CDATA #IMPLIED>
<?xml version="1.0" encoding="utf-8"?>
<time></time>

<!-- Forwards element. -->
<!ELEMENT forwards (forward*)>

<!-- Forward. -->
<!ELEMENT forward EMPTY>
<!ATTLIST forward
type (x11|agent) #REQUIRED
state (on|off|denied) #REQUIRED>

<!-- Compression. -->
<!ELEMENT compression EMPTY>
<!ATTLIST compression
name CDATA #IMPLIED
level CDATA #IMPLIED>

<!-- Authentication-success-message -->
<!ELEMENT authentication-success-message EMPTY>
<!ATTLIST authentication-success-message
enable (yes|no|YES|NO) "yes">

<!-- Quiet-mode -->
<!ELEMENT quiet-mode EMPTY>
<!ATTLIST quiet-mode
enable (yes|no|YES|NO) "no">

<!-- Sftpg3-mode -->
<!ELEMENT sftpg3-mode EMPTY>
<!ATTLIST sftpg3-mode
compatibility-mode CDATA "tectia">

<!-- Terminal-selection -->
<!ELEMENT terminal-selection EMPTY>
<!ATTLIST terminal-selection
selection-type (select-words|select-paths) "select-words">

<!-- Terminal-bell -->
<!ELEMENT terminal-bell EMPTY>
<!ATTLIST terminal-bell
bell-style (none|pc-speaker|system-default) "system-default">

<!-- Close-window-on-disconnect -->
<!ELEMENT close-window-on-disconnect EMPTY>
<!ATTLIST close-window-on-disconnect
enable (yes|no) "no">

<!-- Checksum -->
<!ELEMENT checksum EMPTY>
<!ATTLIST checksum
type (yes|no|md5|sha1|md5-force|sha1-force|checkpoint|
YES|NO|MD5|SHA1|MD5-FORCE|SHA1-FORCE|CHECKPOINT) "yes">

<!-- User-config-directory -->
<!ELEMENT user-config-directory EMPTY>
<!ATTLIST user-config-directory>
<!ELEMENT file-access-control   EMPTY>
<!ATTLIST file-access-control
  enable                    (yes|no|YES|NO) "no">

<!ELEMENT address-family        EMPTY>
<!ATTLIST address-family
  type                      (any|inet|inet6)
  "&default-address-family-type;">

<!ELEMENT protocol-parameters   EMPTY>
<!ATTLIST protocol-parameters
  threads                   CDATA           #IMPLIED>

<!ELEMENT profiles              (profile*)>

<!ELEMENT profile               (hostkey|ciphers|macs|kexs|hostkey-algorithms|
  transport-distribution|rekey|
  authentication-methods|
  user-identities|
  compression|proxy|idle-timeout|
  tcp-connect-timeout|keepalive-interval|
  exclusive-connection|server-banners|
  forwards|tunnels|extended|remote-environment|
  server-authentication-methods|password|
  profile-group)*>

<!ATTLIST profile
  id                        CDATA           #IMPLIED
  name                      CDATA           #IMPLIED
  host                      CDATA           #REQUIRED
  port                      CDATA           "22"
  protocol                  CDATA           "secsh2"
  host-type                 (unix|windows|default) "default"
  connect-on-startup        (yes|no|YES|NO) "no"
  user                      CDATA           #IMPLIED
  gateway-profile           CDATA           #IMPLIED>

<!ELEMENT profile-group EMPTY>
<!ELEMENT network   EMPTY>
<!ATTLIST network
id          ID          #REQUIRED
address     CDATA      #IMPLIED
domain      CDATA      #IMPLIED
ip-generate-start  CDATA      #IMPLIED
ip6-generate-start CDATA      #IMPLIED>

<!ELEMENT dns   EMPTY>
<!ATTLIST dns
id          ID          #REQUIRED
network-id  IDREF      #IMPLIED
application CDATA      #IMPLIED
host        CDATA      #IMPLIED
ip-address  CDATA      #IMPLIED
pseudo-ip   (yes|no|YES|NO) "no">}

<!ELEMENT filter   EMPTY>
<!ATTLIST filter
dns-id      IDREF      #REQUIRED
ports       CDATA      #REQUIRED
action      (block|direct|tunnel|ftp-tunnel|ftp-proxy|BLOCK|DIRECT|TUNNEL|FTP-TUNNEL|FTP-PROXY)
            #REQUIRED
profile-id  CDATA      #IMPLIED
destination CDATA      #IMPLIED
destination-port CDATA      #IMPLIED
fallback-to-plain (yes|no|YES|NO) "no">}

<!ELEMENT rule   EMPTY>
<!ATTLIST rule
application  CDATA      #IMPLIED
host        CDATA      #IMPLIED
ip-address  CDATA      #IMPLIED
pseudo-ip   (yes|no|YES|NO) "no"
ports       CDATA      #REQUIRED
action      (block|direct|tunnel|ftp-tunnel|ftp-proxy|BLOCK|DIRECT|TUNNEL|FTP-TUNNEL|FTP-PROXY)
            #REQUIRED
profile-id  CDATA      #IMPLIED
destination CDATA      #IMPLIED
destination-port CDATA      #IMPLIED
username    CDATA      #IMPLIED
hostname-from-app (yes|no|YES|NO) "no"
username-from-app (yes|no|YES|NO) "no"
fallback-to-plain (yes|no|YES|NO) "no"
show-sftp-server-banner (yes|no|YES|NO) "no">}

<!ELEMENT logging   (log-target*,log-events*)>
<!-- Log events. -->
<!-- Log event facility. -->
<!ENTITY default-log-event-facility "normal">

<!-- Log event severity. -->
<!ENTITY default-log-event-severity "notice">

<!ELEMENT log-target EMPTY>
<!ATTLIST log-target
    file                      CDATA                           #IMPLIED
    type                      (file|syslog|socket|discard)    "file"
    format                    (syslog|csv|xml)                "syslog">

<!ELEMENT log-events (log-target|#PCDATA)*>  
<!ATTLIST log-events
    facility      (normal|daemon|user|auth|local0|local1|
                   local2|local3|local4|local5|local6|local7|discard) 
                   "&default-log-event-facility;"
    severity      (informational|notice|warning|error|critical|
                   security-success|security-failure) 
                   "&default-log-event-severity;">
Appendix B Command-Line Tools and Man Pages

Tectia Server for IBM z/OS contains several command-line tools. Their functionality is explained in the following appendices. The same information is also available in manual pages that are installed to /opt/tectia/man.

• ssh-broker-g3(1): Connection Broker – Generation 3
• ssh-broker-ctl(1): Connection Broker control utility
• ??? Connection Broker probe-key host-key utility
• ssh-troubleshoot(8): utility for collecting system information for troubleshooting purposes
• sshg3(1): Secure Shell terminal client – Generation 3
• scpg3(1): Secure Shell file copy client – Generation 3
• sftpg3(1): Secure Shell file transfer client – Generation 3
• ssh-translation-table(1): Secure Shell file transfer translation table
• ssh-sft-stage(1): stage and destage MVS data sets and HFS files
• ssh-keygen-g3(1): authentication key pair generator
• ssh-keydist-g3(1): Tectia key distribution tool
• ssh-keyfetch(1): utility for downloading server host keys
• ssh-cmpclient-g3(1): certificate CMP enrollment client
• ssh-scepclient-g3(1): certificate SCEP enrollment client
• ssh-certview-g3(1): certificate viewer
• ssh-ekview-g3(1): external key viewer
For a description of the Connection Broker and SOCKS Proxy configuration file options, see `ssh-broker-config(5)`.

**Note**

On the man pages, *Unix* is generally used to refer to all Unix-like operating systems, including Unix System Services (USS) of z/OS.
**Synopsis**

```
```

**Note**

The information presented here is also valid for the `ssh-socks-proxy` command. 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. The SOCKS Proxy uses a separate Connection Broker address that is meant only for transparent FTP tunneling and FTP-SFTP conversion. Normal clients (for example, `sshg3`) should not connect to that address.

**Description**

`ssh-broker-g3` is a component of Tectia client tools for z/OS. It handles all cryptographic operations and authentication-related tasks for the Tectia client programs `sshg3`, `scpg3`, and `sftpg3`.

`ssh-broker-g3` uses the Secure Shell version 2 protocol to communicate with a Secure Shell server.

You can start the Connection Broker manually by using the `ssh-broker-g3` command. This starts `ssh-broker-g3` in the background and all following uses of `sshg3`, `sftpg3`, or `scpg3` will connect via this instance of the Connection Broker instead of starting a new Broker session.

If a command-line client (`sshg3`, `sftpg3`, or `scpg3`) is started when the Connection Broker is not running in the background, the client starts the Broker in `run-on-demand` mode. In this mode, `ssh-broker-g3` will exit after the last client has disconnected.

If there is an `ssh-broker-g3` process running in the run-on-demand mode, and the Connection Broker is started from the command line, the new `ssh-broker-g3` process sends a message to the old `ssh-broker-g3` process to change from the run-on-demand mode to the background mode, keeping the Broker running after the clients disconnect.

The status of the running Connection Broker can be checked using the `ssh-broker-ctl` and `ssh-broker-gui` utilities.

**Authentication**

The Connection Broker operates automatically as an authentication agent, storing user’s public keys and forwarding the authentication over Secure Shell connections. Key pairs can be created with `ssh-keygen-g3`.

The Connection Broker can also serve OpenSSH clients as an authentication agent.
The public key pairs used for user authentication are by default stored in the `~/.ssh2` directory. See the section called “Files” for more information.

The Connection Broker automatically maintains and checks a database containing the public host keys used for authenticating Secure Shell servers. When logging in to a server host for the first time, the host's public key is stored in the user's `~/.ssh2/hostkeys` directory. See the section called “Files” for more information.

**Options**

The most important options of `ssh-broker-g3` are the following:

- `-a, --broker-address=ADDR`
  - Listens to Connection Broker connections on a local address `ADDR`.

- `-D, --debug=LEVEL`
  - Sets the debug level string to `LEVEL`.

- `-f, --config-file=FILE`
  - Reads the Connection Broker configuration file from `FILE` instead of the default location.

- `-l, --debug-log-file=FILE`
  - Dumps debug messages to `FILE`.

- `--pid-file=FILE`
  - Stores the process ID of the Connection Broker to `FILE`.

- `--exit`
  - Make the currently running Connection Broker exit. This will terminate all connections.

- `-V, --version`
  - Displays program version and exits.

- `-h, --help`
  - Displays a short summary of command-line options and exits.

**Environment Variables**

In order to run `ssh-broker-g3` the following environment variables must be set:

- `_BPXK_AUTOCVT=ON`
  - If this variable is not set correctly, `ssh-broker-g3` fails to start.

- `_BPXK_SHAREAS=NO`
  - This variable defines that `ssh-broker-g3` and the client processes are run in separate address spaces.
The following optional environment variable is required in certain situations:

**SSH_SECSH_BROKER=ADDRESS**
This variable defines an address to a separate Tectia Connection Broker process to which a connection is made.

This variable becomes necessary to define the location of the Connection Broker process, if you are running it from a non-default location, or using a userID other than that of the ssh-broker-g3 process owner.

The following environment variables are optional:

**SSH_ZCONSOLE_MSGLEVEL=LEVEL**
This variable specifies the amount of WTO messages that appear on the operator console. Possible values are:

- 0: All WTO messages are shown.
- 1: Unsolicited WTO messages are suppressed.
- 2: All WTO messages are suppressed.

If this environment variable is not set, Unsolicited WTO messages are suppressed.

**SSH_ZCONSOLE_ROUTING_CODE=CODE**
This variable specifies the z/OS routing codes of WTO messages. Possible values are 1 to 128. The default value is 1,11

**Files**

ssh-broker-g3 uses the following files:

$HOME/.ssh2/ssh-broker-config.xml
This is the user-specific configuration file used by ssh-broker-g3 (and sshg3, scp3, and sftpg3). The format of this file is described in ssh-broker-config(5). This file does not usually contain any sensitive information, but the recommended permissions are read/write for the user, and not accessible for others.

$HOME/.ssh2/random_seed
This file is used for seeding the random number generator. It contains sensitive data and its permissions should be read/write for the user and not accessible for others. This file is created the first time the program is run and it is updated automatically. You should never need to read or modify this file.

$HOME/.ssh2/identification
This file contains information on public keys and certificates used for user authentication when contacting remote hosts.
With Tectia Client G3, using the identification file is not necessary if all user keys are stored in the default directory and you allow all of them to be used for public-key and/or certificate authentication. If the identification file does not exist, the Connection Broker attempts to use each key found in the $HOME/.ssh2 directory. If the identification file exists, the keys listed in it are attempted first.

The identification file contains a list of private key filenames each preceded by the keyword IdKey (or CertKey). An example file is shown below:

| IdKey   | mykey |

This directs the Connection Broker to use $HOME/.ssh2/mykey when attempting login using public-key authentication.

The files are by default assumed to be in the $HOME/.ssh2 directory, but also a path to the key file can be given. The path can be absolute or relative to the $HOME/.ssh2 directory. If there is more than one IdKey, they are tried in the order that they appear in the identification file.

$HOME/.ssh2/hostkeys

This is the user-specific default directory for storing the public keys of server hosts. You are prompted to accept new or changed keys automatically when you connect to a server, unless you have set strict-host-key-checking to yes in the ssh-broker-config.xml file. You should verify the key fingerprint before accepting new or changed keys.

When the host key is received during the first connection to a remote host (or when the host key has changed) and you choose to save the key, its filename is stored by default in hashed format. The hashed host key format is a security feature to make address harvesting on the hosts difficult.

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

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

If both hashed and plain-text format keys exist, the hashed format takes precedence.

Note that the identification is different based on the host and port the client is connecting to. For example, the short hostname alpha is considered different from the fully qualified domain name alpha.example.com. Also a connection with an IP, for example 10.1.54.1, is considered a different host, as is a connection to the same host but different port, for example alpha.example.com#222.

For more information on host keys, see Section 4.2.

$HOME/.ssh2/hostkeys/salt

This is the initialization file for hashed host key names.
This is the configuration file used by ssh-broker-g3 (and sshg3, scpg3, and sftpg3) that contains the factory default settings. It is not recommended to edit the file, but you can use it to view the default settings. The format of this file is described in ssh-broker-config(5).

This is the global (system-wide) configuration file used by ssh-broker-g3 (and sshg3, scpg3, and sftpg3). The format of this file is described in ssh-broker-config(5).

If a host key is not found in the user-specific $HOME/.ssh2/hostkeys directory, this is the next location to be checked for all users. Host key files are not automatically put here but they have to be updated manually by the system administrator (root).

If the administrator obtains the host keys by connecting to each host, the keys will be by default in the hashed format. In this case, also the administrator's $HOME/.ssh2/hostkeys/salt file has to be copied to the /opt/tectia/etc/hostkeys directory.

This is the initialization file for hashed host key names. The file has to be copied here manually by the same administrator that obtains the host keys.

This is the default system-wide file used by OpenSSH clients for storing the public key data of known server hosts. It is supported also by Tectia client tools for z/OS.

If a host key is not found in the user-specific $HOME/.ssh/known_hosts file, this is the next location to be checked for all users.

The known_hosts file is never automatically updated by Tectia Client or ConnectSecure, since they store new host keys always in the Tectia user-specific directory $HOME/.ssh2/hostkeys.

This is the default user-specific file used by OpenSSH clients for storing the public key data of known server hosts. The known_hosts file is supported also by Tectia client tools for z/OS.

The known_hosts file contains a hashed or plain-text format entry of each known host key and the port used on the server, in case it is non-standard (other than 22). For more information on the format of the known_hosts file, see the OpenSSH sshd(8) man page.

The known_hosts file is never automatically updated by Tectia Client or ConnectSecure, since they store new host keys always in the Tectia directory $HOME/.ssh2/hostkeys.
$HOME/.ssh2/authorized_keys (on the server host)

This directory is the default location used by Tectia Server for the user public keys that are authorized for login.

On Tectia Server on Windows, the default directory for user public keys is %USERPROFILE%\ssh2\authorized_keys.

$HOME/.ssh2/authorization (on the server host)

This is the default file used by earlier versions of Tectia Server (sshd2) that lists the user public keys that are authorized for login. The file can optionally be used with Tectia Server G3 (ssh-server-g3) as well.

On Tectia Server on Windows, the authorization file is by default located in %USERPROFILE%\ssh2\authorization.

For information on the format of this file, see the ssh-server-g3(8) man page.

$HOME/.ssh/authorized_keys (on the server host)

This is the default file used by OpenSSH server (sshd) that contains the user public keys that are authorized for login.

For information on the format of this file, see the OpenSSH sshd(8) man page.
ssh-broker-ctl

ssh-broker-ctl -- Tectia Connection Broker control utility

Synopsis

ssh-broker-ctl command
[options...]

Note

The information presented here is also valid for the ssh-socks-proxy-ctl command. Running ssh-socks-proxy-ctl is otherwise equal to running ssh-broker-ctl, but the command controls the ssh-socks-proxy process instead of the ssh-broker-g3 process. ssh-socks-proxy-ctl locates automatically the Connection Broker address that the ssh-socks-proxy process is using.

Description

ssh-broker-ctl is a control utility for the Connection Broker (ssh-broker-g3). It can be used, for example, to view the status of the Connection Broker, to stop the Connection Broker, to manage keys and certificates, and to manage connections.

Options

The following general options are available:

-a, --broker-address ADDRESS
Defines an address to a separate Tectia Connection Broker process to which a connection is made.

The same effect can be achieved by defining a Connection Broker address with environment variable SSH_SECSH_BROKER.

Tip

If you are running ssh-broker-ctl using a userID other than that of the ssh-broker-g3 process owner, the -a option must be given so that ssh-broker-ctl knows where to connect. In this case, you must also run ssh-broker-ctl as a privileged user (root).

For example, when user SSBRKR owns the ssh-broker-g3 process, run the ssh-broker-ctl with commands:

```
# ssh-broker-ctl -a /tmp/ssh-SSBRKR/ssh-broker status -s
# ssh-broker-ctl -a /tmp/ssh-SSBRKR/ssh-broker status --pid
# ssh-broker-ctl -a /tmp/ssh-SSBRKR/ssh-broker list-connections
```
-D, --debug STR
Defines the debug level.

-e, --charset=CS
Defines the character set to be used in the output. The supported character sets are utf8, iso-8859-1, latin1, iso-8859-15, latin9, and ascii.

-q, --quiet
Defines that little or no output is to be displayed, depending on the command.

-s, --short
Defines that a shorter, more machine readable, output format is to be used.

--time-format=FMT
Defines the time format to be used in the output. The default depends on the system locale settings.

-v, --verbose
Defines that more information, if available, is to be output.

-V, --version
Displays the version string.

-w, --wide
Defines that the output will not be truncated, even if it means long lines.

-h, --help
Displays a context-sensitive help text on command-line options. Help is available also on specific commands. For example, to get help on the status command, run:

$ ssh-broker-ctl status --help

Commands

Note
For a detailed description of the command options, use the command-specific --help option.

ssh-broker-ctl accepts the following commands:

add-certificate [options] <certificate-file>
Adds the given X.509 sub-CA certificate to the Connection Broker certificate cache. The certificate can be used in certificate validations but it is not stored permanently. Restarting the Connection Broker will remove the certificate.

add-crl [options] <crl-file>
Adds the given X.509 CRL to the Connection Broker CRL cache. The CRL can be used in certificate validations but it is not stored permanently. Restarting the Connection Broker will remove the CRL.
add-key filename
   Adds a new private key from the given file name. The private key is not stored permanently in the configuration. Stopping the Connection Broker will remove the key.

add-provider type parameter
   Registers a key provider to the Connection Broker. The type option is one of the supported provider types and the parameter option is a parameter string specific to the provider type.

   For a list of the supported key provider types and the corresponding parameter formats, use the command-specific --help option.

auth-handler [options]
   Registers itself as the default authentication form handler. All authentication prompts for clients that are unable to handle them (mostly SOCKS proxy and other tunnels) are directed to this client.

   For a list of the supported key provider types and the corresponding parameter formats, use the command-specific --help option.

close-channel channel-id ...
   Closes the defined channel. You can also enter multiple channel-IDs to close several channels.

close-connection connection-id ...
   Closes the defined connection. You can also enter multiple connection-IDs to close several connections.

close-tunnel-listener tunnel-id ...
   Closes open tunnel listener. Tunnel id is either the id number returned by ssh-broker-ctl list-tunnel-listeners command or a listen address and port pair separated by a colon. If the listen address is omitted, local listeners (127.0.0.1) are selected. As an example, the following command closes the listener with id 7, and the ones listening at 168.192.0.15 port 1234 and 127.0.0.1 port 2112:

   $ ssh-broker-ctlctl 7 168.192.0.15:1234 :2112

config-value [options] path
   Retrieves configuration values from the Connection Broker based on the defined path and displays them.

connection-status [--show-channels][--write-hostkey=FILE] connection-id
   Displays a detailed connection status for the connection ID (the numeric identifier shown by the list-connections) command.

connector [options] [enable|disable]
   Enables and disables the Connector functionality in the Connection Broker. Without parameters prints the current state.

disconnect-client client-id
   Disconnects a Connection Broker client process.

Sets the Connection Broker debug level to the defined level. If no debug-level parameter is given here, the current debug level is not changed.

keylog [--remove] [--all] [--update <key-id|key-hash>] [--init] [--uninit] [--close] [-v, --verbose] [key-id|key-hash|hostname]

Keylog is used to manage uploaded public keys and to display a log of them. The Keylog does not store the public keys, it only stores information about the keys and the hosts where the keys have been uploaded to. The information can be used to manage the keys at a later stage, for example, to track hosts where a key has been uploaded to. The keylog is not on by default, it must be enabled first.

Without the options, displays a list of the uploaded keys. If a key or a hostname is specified, only the selected keys are displayed.

key-passphrase[--all][--clear][--passphrase-file=filename][--passphrase-string=passphrase] [key-id|key-hash]

Prompts the user's private-key passphrase or PIN code.

key-upload [options] key [user@]server[#port]

Uploads the selected key (key can be a key ID number, a public key hash or a file name) into the authorized keys directory or file on the server, depending on the automatically detected upload method. After the operation, the key can be used in public-key authencation to log in to the server without a password. If the keylog is enabled, the command prompts for a keylog passphrase (if needed), and information about the public keys is stored in the key upload log.

list-connections [-c, --show-channels] [-s, --short] [--client-pid=PID] [--disconnected]

Displays a list of the currently open connections together with connection parameters and traffic statistics. Displays also the connection ID which can be used with other commands to identify the connection.

list-channels [-s, --short]

Displays a list of the currently open connection channels, together with channel type and traffic statistics. Displays also the channel ID which is used by other commands to identify the connection.

list-clients [-c, --show-channels] [-s, --short] [--all]

Displays a list of the currently connected client processes.

list-keys [-s, --short] [--extra certificates] [--provider-ID]

Displays a list of the user's private keys, together with the basic key attributes such as the key type, size, and possible file name or key provider information. Outputs also the fingerprint and the identifier of the key. The identifier is used by other Connection Broker commands to identify the private key.

list-profiles [-s, --short] [-v, --verbose] [ name ...]

Displays a list of connection profiles in the Connection Broker. Shows the profile name and basic connection settings, such as the host and the user name. If profile names are given, only those profiles are listed.
list-providers [ provider ... ]
Displays a list of the key providers in the Connection Broker. If one or more provider names or ID numbers are given, only those providers will be listed. The provider name can be either a full provider name or a prefix.

list-tunnel-listeners [options]
Displays a list of the currently active tunnel listeners (also called port forwards).

open-tunnel-listener [options] listen-port [user@]server [#port] [dst-host] [dst-port]
Opens a tunnel listener, similar to sshg3 -L and -R options. The difference is that ssh-broker-ctl will exit after the tunnel is opened. The tunnel status can be viewed with ssh-broker-ctl list-tunnel-listeners command and the tunnel can be closed with ssh-broker-ctl close-tunnel-listener command.

In local mode (default), the listener is opened to localhost listen-port. All connections will be tunneled through server and from there to the final destination address and port. Tunnel types socks and socks-proxy do not require destination information as it will be obtained from SOCKS client. Tunnel types tcp, ftp and local require destination information.

pkcs10-sign [options] key-id [subject-name]
Signs a PKCS#10 certificate request with the given key. The key-id can be either a key id or a key hash. The subject name parameter is required unless the template option is used. If the subject name is not a valid distinguished name, it will be wrapped automatically into a common name component. For example, a subject name string My Name will be converted to CN=My Name.

Probes for a Secure Shell server hostkey. Connects to the given address and port (defaults to 22) and displays the server's public key or certificate.

For more details about the available options, see ???.

remove-key [options] key-id
Removes a private key permanently.

remove-provider [--all] provider-id
Removes a key provider from the Connection Broker.

start
Starts the Connection Broker in daemon mode if it is not already running.

start-gui
Starts the Connection Broker GUI process unless it is already running.

status [-s, --short] [-q, --quiet] [-p id] [-a]
Without parameters, displays short statistics and a configuration summary for the currently running Connection Broker process.
stop
   Stops the Connection Broker.

validate-certificate [options] <certificate-file>
   Validates the given X.509 certificate. If a host name is given, also checks if the certificate would be accepted as a host certificate for the host.

view-key [-s, --short] [-v, --verbose] [--clear] [--write-key= file] key-id
   Displays information on the defined key. If the key has certificates, a short summary of them is also shown.

Environment Variables

In order to run ssh-broker-ctl the following environment variables must be set:

_BPXK_AUTOCVT=ON
   If this variable is not set correctly ssh-broker-ctl fails to start.
ssh-broker-ctl probe-key

ssh-broker-ctl probe-key -- Tectia Connection Broker probe-key host-key utility

Synopsis

ssh-broker-ctl probe-key [options...] host

Description

probe-key is a ssh-broker-ctl command for downloading server host keys, and for optionally setting them as known host keys for the Secure Shell client. It is typically used by the system administrator during the initial setup phase. probe-key is intended to be a preferred method over ssh-keyfetch for managing host keys and it supports the same crypto algorithms as the Secure Shell client.

Options

The following general options are available:

--add-hostkey

If host key is found, automatically add host key as trusted.

⚠️ Caution

When using either --add-hostkey or --save-hostkey options, received host keys are accepted automatically. To avoid potential man-in-the-middle attacks, we recommend verifying the key fingerprints before accepting them.

To only accept verified keys, obtain the expected host-key fingerprint from a trusted source (for example by calling the server administrator), and specify the expected host-key fingerprint with the --hostkey-fp option.

--save-hostkey

If host key is found, automatically add host key as trusted, and remove any old host keys associated with the host.

⚠️ Caution

When using either --add-hostkey or --save-hostkey options, received host keys are accepted automatically. To avoid potential man-in-the-middle attacks, we recommend verifying the key fingerprints before accepting them.
To only accept verified keys, obtain the expected host-key fingerprint from a trusted source (for example by calling the server administrator), and specify the expected host-key fingerprint with the `--hostkey-fp` option.

`--hostkey-fp=fingerprint`

Save or add host key only if it matches the fingerprint. Cannot be used with `--write-hostkey`.

`--write-hostkey=filename`

Writes a public key or a certificate into a file.

**Caution**

If `--write-hostkey=filename` is specified in the format:

`$home/.ssh2/hostkeys/key_port_hostname.pub`

Then the host key is automatically accepted and all other options are ignored.
ssh-troubleshoot

ssh-troubleshoot -- tool for collecting system information

Synopsis

ssh-troubleshoot [options] [command [command-options]]

Description

ssh-troubleshoot is a tool for collecting information on the operating system (its version, patches, configuration settings, installed software components, and the current environment and state) and on the Tectia installation (installed product components and versions, their state, and the global and user-specific configurations).

The collected information will be stored in a file named ssh_troubleshoot_<host>_<date>_<time>.tar. Send the file to the SSH technical support for analysis to help in troubleshooting situations.

To get all necessary information, run the command as an administrator, because it might need root access to some directories.

The default location of this tool is /opt/tectia/sbin/, and that path should be set in the user's PATH. Otherwise ssh-troubleshoot can be run only by pointing directly to the executable file. In case you want to install Tectia tools to any non-default directory, for example to /usr/lpp/tectia/, you must set that directory in the user's PATH.

Options

Enter each option separately, they cannot be combined. The following options are available:

- \(d\), --debug LEVEL
  
  Sets the debug level string to LEVEL.

- \(k\), --keep-going
  
  Defines that the data collecting is continued as long as possible, even after errors are encountered.

- \(o\), --output FILENAME
  
  Defines a non-default output file for storing the collected data.

  If FILENAME is '-', the collected data is added to the standard output. The default output file is created in a temporary archive directory and stored as ssh-troubleshoot-data-<hostname>-<timestamp>.tar. The timestamp is in format: yyyymmdd-hhmmUTC.

- \(u\), --user USERNAME
  
  Defines another user for the info command, the default is the current user. This affects the home directory from which the user-specific Tectia configuration files are fetched.
-q, --quiet
    Suppresses detailed reporting about the command progress, reports only errors.

-h, --help
    Displays this help text.

Commands

**ssh-troubleshoot** accepts the following command:

```
info
```
Gathers information about the system configuration. The collected data will be stored as a *tar* file.

Options:

```
--include-private-keys
```
Collects everything from the specified user's configuration directories, including the private keys.

By default, the private keys nor unrecognized files are not included in the result data.
sshg3

sshg3 -- Secure Shell terminal client - Generation 3

Synopsis

sshg3 [options...]
profile [user@] host [#port]
[command]

Description

sshg3 is a program for logging in to a remote machine and executing commands on a remote machine. sshg3 provides secure, encrypted communication channels between two hosts over an unsecured network. It can be used to replace the unsecured rlogin, rsh, and telnet programs. Also X11 connections and arbitrary TCP/IP ports can be forwarded over secure channels with sshg3.

To connect to a remote host using sshg3, give either the name of a connection profile defined in the ssh-broker-config.xml file (profile) or the IP address or DNS name of the remote host, optionally with the remote user name and the port of the Secure Shell server ([user@]host[#port]). If no user name is given, the local user name is assumed. If no port is given, the default Secure Shell port 22 is assumed. The remote host must be running a Secure Shell version 2 server.

sshg3 acts as a Connection Broker client and launches the actual Connection Broker process, ssh-broker-g3 as a transport (in run-on-demand mode), or uses an already running Connection Broker process. The Connection Broker will ask the user to enter a password or a passphrase if they are needed for authentication. Connection Broker uses the configuration specified in the ssh-broker-config.xml file.

When the user's identity has been accepted by the server, the server either executes the given command, or logs in to the machine and gives the user a normal shell. All communication with the remote command or shell will be automatically encrypted.

If no pseudo-tty has been allocated, the session is transparent and can be used to securely transfer binary data.

The session terminates when the command or shell on the remote machine exits and all X11 and TCP/IP connections have been closed. The exit status of the remote program is returned as the exit status of sshg3.

Agent Forwarding

ssh-broker-g3 acts as an authentication agent, and the connection to the agent is automatically forwarded to the remote side unless disabled in the ssh-broker-config.xml file or on the sshg3 command line (with the -a option).
TCP Port Forwarding

Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either in the `ssh-broker-config.xml` file or on the `sshg3` command line (with the `-L` and `-R` options).

Options

Command-line options override the settings in the `ssh-broker-config.xml` file if the same option has been configured in both places. The following options are available:

- `-a`, `--no-agent-forwarding`
  Disables authentication agent forwarding. In the factory settings, agent forwarding is enabled.

+ `a`
  Enables authentication agent forwarding. In the factory settings, agent forwarding is enabled, but it can be denied in the Connection Broker configuration file, in which case users cannot enable it on the command-line and this `+a` will be ignored.

- `-B`, `--batch-mode`
  Uses batch mode. Fails authentication if it requires user interaction on the terminal.

  Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

- `-c`, `--ciphers=LIST`
  Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

  ```
  --ciphers seed-cbc@ssh.com,aes256-cbc
  ```

  Enter `help` as the value to view the currently supported cipher names.

- `-D`, `--debug=LEVEL`
  Sets the debug level. `LEVEL` is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

**Note**

The debug level can be set only when the `sshg3` command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.
-e, --escape-char=CHAR
Sets escape character (none: disabled, default: ~).

-g, --gateway
Gateways ports, which means that also other hosts may connect to locally forwarded ports. This option
has to be specified before the "-l" option. Note the logic of + and - in this option.

+g
Does not gateway ports. Listens to tunneling connections originating only from the localhost. This is the
default value. Note the logic of + and - in this option.

-i FILE
Defines that private keys defined in the identification file are used for public-key authentication.

-K, --identity-key-file=FILE
Defines that the given key file of a private key or certificate is used in user authentication. The path to
the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection
Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If
the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the
certificate and the private key can be given using multiple -K options on command line.

-L, --localfwd [protocol/][listen-address:]listen-port:dst-host:dst-port
Forwards a port on the local (client) host to a remote destination host and port.

This allocates a listener port (listen-port) on the local client. Whenever a connection is made to this
listener, the connection is tunneled over Secure Shell to the remote server and another connection is made
from the server to a specified destination host and port (dst-host:dst-port). The connection from the
server onwards will not be secure, it is a normal TCP connection.

Giving the argument protocol enables protocol-specific forwarding. The protocols implemented are
tcp (default, no special processing), ftp (temporary forwarding is created for FTP data channels, effect-
ively securing the whole FTP session), and socks.

With the socks protocol, the syntax of the argument is "-L socks/[listen-address:]listen-port".
When this is set, Tectia Client or ConnectSecure will act as a SOCKS server for other applications, cre-
ating forwards as requested by the SOCKS transaction. This supports both SOCKS4 and SOCKS5.

If listen-address is given, only that interface on the client is listened. If it is omitted, all interfaces are
listened.

-l, --user=USERNAME
Logs in using this user name.
-m, --macs=LIST
Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

```
--macs hmac-sha1-96,hmac-md5,hmac-md5-96
```

Enter help as the value to view the currently supported MAC names.

-u, --kexs=kexs
Sets the allowed key exchange (KEX) methods to be offered to the server. List the KEX names in a comma-separated list. For example:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

Enter help as the value to view the currently supported KEX methods.

-j, --hostkey-algs=algs
Sets the allowed host key algorithms to be offered to the server. List the host key algorithms in a comma-separated list. For example:

```
--hostkey-algs ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com
```

Enter help as the value to view the currently supported host key algorithms.

-n, --dev-null
Redirects input from /dev/null.

-o option
Processes an option as if it was read from a Tectia Client 4.x-style configuration file. The supported options are ForwardX11, ForwardAgent, AllowedAuthentications and PidFile. For example, -o "ForwardX11=yes". Also -o "PidFile=/tmp/sshg3.pid" makes sshg3 to store its process ID into file "/tmp/sshg3.pid" if it goes into background.

-P, --password= PASSWORD | file://PASSWORDFILE | extprog://PROGRAM
Sets user password that the client will send as a response to password authentication. The PASSWORD can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (--password-file://PASSWORDFILE), or entering a path to a program or script that outputs the password (--password=extprog://PROGRAM).

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

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

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

- **-p, --port=PORT**
  Connects to this port on the remote host. A Secure Shell server must be listening on the same port.

- **-q**
  Quiet mode, reports only fatal errors. This option overrides the `quiet-mode` setting made in the Connection Broker configuration file.

- **-R, --remotefwd [protocol/] [listen-address:] listen-port:dst-host:dst-port**
  Forwards a port on the remote (server) host to a destination host and port on the local side.

  This allocates a listener port (`listen-port`) on the remote server. Whenever a connection is made to this listener, the connection is tunneled over Secure Shell to the local client and another connection is made from the client to a specified destination host and port (`dst-host:dst-port`). The connection from the client onwards will not be secure, it is a normal TCP connection.

  Giving the argument `protocol` enables protocol-specific forwarding. The protocols implemented are `tcp` (default, no special processing) and `ftp` (temporary forwarding is created for FTP data channels, effectively securing the whole FTP session).

  If `listen-address` is given, only that interface on the server is listened. If it is omitted, all interfaces are listened.

- **-S, --no-session-channel**
  Does not request a session channel. This can be used with port-forwarding requests if a session channel (and tty) is not needed, or the server does not give one.

- **+S**
  Requests a session channel. This is the default value.

- **-s, --subsystem subsystem remote_server**
  Sets a subsystem or a service to be invoked on the remote server. The subsystem is specified as a remote command. For example: `sshg3 -s sftp <server>`

- **-t, --tty**
  Allocates a tty even if a command is given.

- **-v, --verbose**
  Uses verbose mode. More information or error diagnostics are output if a connection fails.
-z, --broker-log-file=FILE
Sets the Connection Broker log file to FILE. This option works only if ssh-broker-g3 gets started by this process.

--aa, --allowed-authentications=METHODS
Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. For example:

```
--allowed-authentications keyboard-interactive,password
```

Enter help as the value to view the currently supported authentication methods.

--abort-on-failing-tunnel
Aborts if creating a tunnel listener fails (for example, if the port is already reserved).

--charset-conversion=FORCE|NO
Specifies whether character set conversion is to be performed. The default is FORCE.

--compressions=METHODS
Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter help as the value to view the currently supported compression methods.

--exclusive
Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

--hostkey-policy=POLICY
Defines the policy for checking server host keys and handling unknown server host keys. The possible values are:

- ask (default): The user will be asked to verify and accept the server host keys, if the keys are not found in the host key storage or if the keys have changed.

- strict: The connection to the server will be allowed only if the host key is found in the user's known host keys storage.

- tofu: Trust on first use; new host keys are stored without prompting the user to accept them.

- advisory (not recommended): New host keys are stored without prompting the user to accept them, and connections are allowed also to servers offering a changed host key.

⚠️ Caution
Consider carefully before setting the policy to advisory. Disabling the host-key checks makes the connection vulnerable to attacks.
You can also configure the host key policy in the `ssh-broker-config.xml` configuration file with the `<auth-server-publickey>` element in the `default-settings` and per profile. See `auth-server-publickey`.

If this option is set on the command-line client and configured in the `ssh-broker-config.xml`, the command-line value will be used.

```
--identity=ID
```

Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

```
--identity-key-hash ID
```

Defines the private key used in user authentication with the corresponding public key hash.

```
--identity-key-id ID
```

Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

```
--keep-alive=VALUE
```

Defines how often keep-alive messages are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

```
--kip
```

Defines keyboard-interactive and password as the allowed methods for user authentication; the same as

```
--allowed-authentications keyboard-interactive,password
```

```
--local-charset=VALUE
```

The default is the character set specified in the user's (local) locale. If that is not available, the default is IBM-1047.

```
--remote-charset=VALUE
```

Defines the character set used on the remote end. The default is UTF8.

```
--remote-environment name=VALUE
```

When this option is used, the defined environment variables are passed to the server from the client side. The environment variables are applied on the server when requesting a command, shell or subsystem.

Note that the server can restrict the setting of environment variables.

You can also configure the environment variables to be passed to the server in the `ssh-broker-config.xml` configuration file with the `<remote-environment>` element in the `default-settings` and per profile. See `remote-environment`.

If the same variable is entered on the command-line client and configured in the `ssh-broker-config.xml`, the command-line version will be used.
--remote-environment-format name=VALUE
The defined environment variables are passed to the server from the client side. The Connection Broker
processes the value before sending it to the server.

You can use %U in the value to indicate a user name. The Connection Broker replaces the %U with the
actual user name before sending it to the server.

For more information, see the --remote-environment option above.

--tcp-connect-timeout=VALUE
Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter
the value as a positive number.

-V, --version
Displays program version and exits.

-h, --help, -?
Displays a short summary of command-line options and exits.

Commands

sshg3 can take as a command either of the following ones:

remote_command [arguments] ...
Runs the command on a remote host.

-s service
Enables a service in remote server.

Escape Sequences

sshg3 supports escape sequences to manage a running session. For an escape sequence to take effect, it must
be typed directly after a newline character (press Enter first). The escape sequences are not displayed on
screen during typing.

The following escape sequences are supported:

~.
Terminates the connection.

~Ctrl-Z
Suspends the session.

~~
Sends the escape character literally.
Lists forwarded connections.

Disables the escape character irrevocably.

Displays a summary of escape sequences.

Initiates rekeying manually.

Gives connection statistics, including server and client version, packets in, packets out, compression, key exchange algorithms, public-key algorithms, and symmetric ciphers.

Uploads the chosen public key automatically to the server. If the user has only one key, it will be uploaded. Otherwise the largest key with a name that matches `id_rsa_<size>_a` will be selected.

Uploads a public key to the server. A list of available keys is printed and the user is prompted to select one to be uploaded.

Gives statistics for individual channels (data window sizes etc). This is for debugging purposes.

Dumps the client version number to stderr (useful for troubleshooting).

Environment Variables

In order to run `sshg3` the following environment variables must be set:

BPXK_AUTOCVT=ON
  If this variable is not set correctly `sshg3` fails to start.

BPXK_BATCH_UMASK=0022
  This variable defines the permissions for newly created files.

BPXK_SHAREAS=NO
  This variable defines that `ssh-broker-g3` and `sshg3` processes are run in separate address spaces.

Upon connection, the Secure Shell server will automatically set a number of environment variables that can be used by `sshg3`. The exact variables set depend on the Secure Shell server. The following variables can be used by `sshg3`:  

© 2007–2020 SSH Communications Security Corporation
HOME
The user's home directory.

LOGNAME
Synonym for user; set for compatibility with systems using this variable.

MAIL
The user's mailbox.

PATH
Set to the default PATH, depending on the operating system or, on some systems, /etc/environment or /etc/default/login.

SSH.SOCKS_SERVER
The address of the SOCKS server used by sshg3.

SSH.ZCPU.MATH.FACILITY
Controls the z13 Vector Facility usage. The default value is 1. If this variable is set to 1, sshg3 will probe the availability of the z13 Vector Facility. If the facility is available, sshg3 will use it. If this variable is set to 0 or the facility is not available, sshg3 will not use the z13 vector facility.

SSH2.AUTH.SOCK
If this exists, it is used to indicate the path of a Unix-domain socket used to communicate with the authentication agent (or its local representative).

SSH2.CLIENT
Identifies the client end of the connection. The variable contains three space-separated values: client IP address, client port number, and server port number.

SSH2.ORIGINAL.COMMAND
This will be the original command given to sshg3 if a forced command is run. It can be used, for example, to fetch arguments from the other end. This does not have to be a real command, it can be the name of a file, device, parameters or anything else.

SSH2.TTY
This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

TZ
The time-zone variable is set to indicate the present time zone if it was set when the server was started (the server passes the value to new connections).

USER
The name of the user.

For a list of variables set by Tectia Server, see the ssh-server-g3(8) man page.
Exit Values

sshg3 returns the following values based on the result of the operation:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation was successful.</td>
</tr>
<tr>
<td>1</td>
<td>sshg3 has encountered an error, the reason is usually given in an error message.</td>
</tr>
</tbody>
</table>

When executing remote commands, sshg3 exits with the status of the command run indicated with exit codes:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The remote command was run successfully.</td>
</tr>
<tr>
<td>127</td>
<td>The requested remote command was not found.</td>
</tr>
</tbody>
</table>

**Note**

When the command is run from JCL using BPXBATCH, the exit values are multiplied by 256.

Examples

Connect as the local user name to host remotehost, port 2222, and open shell:

```bash
$ sshg3 remotehost#2222
```

Connect to the host specified by the connection profile `profile1` in the `ssh-broker-config.xml` file, and run the `who` command (and exit after running the command):

```bash
$ sshg3 profile1 who
```

Connect as `user` to host `remotehost`, and open a local port forwarding from port 143 on the client to port 143 on `imapserver`. Do not open shell. Also other hosts may connect to the local port. The connection from `remotehost` to `imapserver` will not be secured:

```bash
$ sshg3 -L 143:imapserver:143 -g -S user@remotehost
```
**scpg3**

scpg3 -- Secure Shell file copy client - Generation 3

**Synopsis**

```
scpg3 [options...]
[src_profile: | [user@] src_host [#port: | src_file...  
[dst_profile: | [user@] dst_host [#port: | dst_file_or_dir]
```

**Description**

scpg3 is used to securely copy files over the network. scpg3 launches ssh-broker-g3 to provide a secure transport using the Secure Shell version 2 protocol. ssh-broker-g3 will ask for passwords or passphrases if they are needed for authentication. scpg3 uses the configuration specified in the ssh-broker-config.xml file.

Copies between two remote hosts are permitted. The remote host(s) must be running a Secure Shell version 2 server with the sftp-server (or sft-server-g3) subsystem enabled. Tectia Server has sft-server-g3 enabled by default.

Any filename may contain a host, user, and port specification to indicate that the file is to be copied to or from that host ([user@] host [#port]). If no user name is given, the local user name is assumed. If no port is given, the default Secure Shell port 22 is assumed. Alternatively, a connection profile defined in the ssh-broker-config.xml file (profile) can be given.

**Note**

When entering a connection profile in the scpg3 command, note that Tectia client tools for z/OS deduces the meaning of the argument differently depending on its format. If there is an @ sign in the given attribute value, Tectia client tools for z/OS always interprets it to be <username@hostname>, i.e. not a profile.

Also, if there are dots in a profile name (for example host.x.example.com, the dots need to be escaped on command line. Enter host\.x\.example\.com, instead. Otherwise the profile name is taken as a host name and the current Windows user name is used for logging in.

The host parameter can optionally be enclosed in square brackets ([[]]) to allow the use of semicolons. The file argument can contain simple wildcards: asterisk (*) for any number of any characters, and question mark (?) for any one character.

For information on special characters in file names, see the section called “Filename Support”.

© 2007–2020 SSH Communications Security Corporation

Tectia® Server 6.6 for IBM z/OS User Manual
Options

The following command-line parameters can be used to further specify the `scp3` options.

-4

Defines that all connection-related DNS resolutions will be resolved as an IPv4 address.

-6

Defines that all connection-related DNS resolutions will be resolved as an IPv6 address.

-a [arg]

Transfers files using the ASCII mode, that is, newlines will be converted on the fly. For transfers between Tectia on z/OS and other hosts, this also enables automatic ASCII-EBCDIC conversions. See the `sftp3 ascii` command in the section called “Commands”.

If the server does not advertise the newline convention, and you are not using a host profile that specifies its host type, you can give `scp3` a hint by giving an argument after `-a`. The default is to set the destination newline convention, but you can specify either one by prefixing the argument with `src:` or `dest:` for source or destination convention, respectively. The available conventions are dos, unix, and mac, using `\r\n`, `\n`, and `\r` as newlines, respectively. Note that there is no space between the `-a` and its argument.

An example is shown below:

```bash
$ scp3 -asrc:unix -adest:dos src_host:src_file dest_host:dest_file
```

To force the newline conventions, use these values: `force-dos`, `force-unix`, and `force-mac`. These settings force the newline mode irrespective of what the remote SSH server suggests to the SCP client.

-B, --batch-mode

Uses batch mode. Fails authentication if it requires user interaction on the terminal.

Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

-b buffer_size_bytes

Defines the maximum buffer size for one SFTP protocol read or write request (default: 32768 bytes).

The maximum number of SFTP protocol read or write requests sent in parallel within the transfer of a single file can be specified with the `-N` option.

Note that when streaming (see `--streaming`) is used (as it is by default when transferring files larger than `buffer_size_bytes` to/from Tectia Server), this option is not used for defining buffer sizes.

-C

Disables compression from the current connection.
+C
Enables zlib compression for this particular connection.

-c, --ciphers=LIST
Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

Enter `help` as the value to view the currently supported cipher names.

-D, --debug=LEVEL
Sets the debug level. `LEVEL` is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

**Note**
The debug level can be set only when the `scp3` command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

-d
Forces target to be a directory.

-I, --interactive
Prompts whether to overwrite an existing destination file (does not work with `-B`).

-`i` `FILE`
Defines that private keys defined in the identification file are used for public-key authentication.

-K, --identity-key-file=FILE
Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple `-K` options on command line.

-N `max_requests`
Defines the maximum number of SFTP protocol read or write requests sent in parallel (default: 10).

The size of the buffer used in each read or write request can be specified with the `-b` option.

Note that this value applies within the transfer of a single file; it cannot be used to define the number of files sent in parallel.

When streaming (see `--streaming`) is used (as it is by default when transferring files larger than `buffer_size_bytes` specified with the `-b` option to/from Tectia Server), this option is not used.
-O, --offset=r<offset> w<offset> l<length> t<length>
Sets offset. Offset r<offset> specifies the start offset in the source file. Offset w<offset> specifies the start offset in the destination file. Length l<length> specifies the amount of data to be copied. Truncate length t<length>, if given, specifies the length to which the destination file is truncated or expanded after the file data has been copied.

-p
Preserves the file permissions and the timestamps when both the source and the destination are on Unix file systems (including z/OS USS). Preserves the timestamps but not the file permissions, if either one, the source or the destination is on Windows. If the destination is on z/OS MVS, none will be preserved.

-P port
Connects to this Secure Shell port on the remote machine (default: 22).

-Q
Does not show progress indicator. The effect of this option is the same as using --progress-display=no.

Do not use this option together with parameter --statistics.

-q
Uses quiet mode (only fatal errors are shown). This option overrides the quiet-mode setting made in the Connection Broker configuration file.

-r
Recurses subdirectories.

-u, --unlink-source
Removes source files after copying (file move).

-v, --verbose
Uses verbose mode (equal to -D 2).

-W, --whole-file
Does not try incremental checks. By default (if this option is not given), incremental checks are tried. This option can only be used together with the --checksum option.

--aa, --allowed-authentications=METHODS
Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. For example:

```
--allowed-authentications keyboard-interactive,password
```
Enter help as the value to view the currently supported authentication methods.

--append
Appends data to the end of the destination file.
=binary
Uses binary transfer mode. If the server is Tectia Server for IBM z/OS, the server is requested not to perform ASCII to EBCDIC conversion, and the file is transferred using the Stream format. You can use the --src-site and --dst-site options to change the values.

--checkpoint=b <bytes>
Byte interval between checkpoint updates (default: 10 MB). This option can only be used when --checksum=checkpoint.

--checkpoint=s <seconds>
Time interval between checkpoint updates (default: 10 seconds). This option can only be used when --checksum=checkpoint.

--checksum [ -yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]
Uses MD5 or SHA-1 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than buffer_size_bytes are not checked. Use md5-force or sha1-force with small files (default: yes, i.e. use SHA1 checksums in FIPS mode, MD5 checksums otherwise). Use checkpointing when transferring large files one by one.

--compressions=METHODS
Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

Enter help as the value to view the currently supported compression methods.

--dst-site=PARAM
Uses the specified site parameters with the destination files. See the sftpg3 site command in the section called “Commands”.

--exclusive
Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

--fips
Performs the checksums using the FIPS cryptographic library.

--force-lower-case
Destination filename will be converted to lowercase characters.

--hostkey-algorithms=HOSTKEYALGORITHMS
Sets the allowed host key algorithms to be offered to the server. List the host key algorithms in a comma-separated list. For example:

--hostkey-algorithms ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com

Enter help as the value to view the currently supported host key algorithms.
--overwrite [ -yes | no ]
Selects whether to overwrite existing destination file(s) (default: yes).

--identity=ID
Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

--identity-key-hash=ID
Defines the private key used in user authentication with the corresponding public key hash.

--identity-key-id=ID
Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

--keep-alive=VALUE
Defines how often keep-alive messages (non-operation packages) are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.

--kexs=kexs
Sets the allowed key exchange (KEX) methods to be offered to the server. List the KEX names in a comma-separated list. For example:

`--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com`

Enter help as the value to view the currently supported KEX methods.

--kip
Defines keyboard-interactive and password as the allowed methods for user authentication; the same as

`--allowed-authentications keyboard-interactive,password`

--macs=LIST
Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

`--macs hmac-sha1-96,hmac-md5,hmac-md5-96`

Enter help as the value to view the currently supported MAC names.

--password= PASSWORD | file://PASSWORDFILE | extprog://PROGRAM
Sets user password that the client will send as a response to password authentication. The PASSWORD can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (--password-file://PASSWORDFILE), or entering a path to a program or script that outputs the password (--password-extprog://PROGRAM).

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

`© 2007–2020 SSH Communications Security Corporation`
#!/usr/bash

cat /full/pathname/to/my_password.txt

⚠️ Caution

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.

`--prefix=PREFIX`

Adds a prefix to a filename during the file transfer. The prefix is removed after the file has been successfully transferred.

On z/OS, when applied to MVS data set names, the prefix is inserted after the High Level Qualifier (HLQ). In case you want the prefix to be a separate qualifier, include a dot at the end of the prefix:

`--prefix=PREFIX.`

`--src-site=PARAM`

Uses the specified site parameters with the source files. See the `site` command in the section called “Commands”.

`--statistics [ -no | yes | simple ]`

ℹ️ Note

In release 6.1.5, the behavior of the `--statistics` option has changed and the `--statistics-format` option has been removed. Instead of them, use the new `--summary-display` and `--summary-format` options.

The `--statistics` option chooses the style of the statistics to be shown after a file transfer operation. Note that `--statistics` and `--summary-display` must not be used together.

The `--statistics` option takes the following values:

- **no** - no statistics will be created.
- **yes** - shows a progress bar during the file transfer. This is the default. An example of the output:

  ```
  scp3 --statistics="yes" sourcefile destinationfile
  sourcefile                     |  127MB |  42.9MiB/s | TOC: 00:00:03 | 100%
  ```

- **simple** - simple one-line statistics will be displayed after the file transfer has ended. For example:

  ```
  scp3 --statistics=simple sourcefile destinationfile
  sourcefile       |  127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
  ```
--summary-display [ -no | yes | simple | bytes ]

Chooses the style of the file transfer summary data to be displayed after a file transfer operation. With the summary display, the progress bar data is also displayed by default.

Note that --summary-display and --statistics must not be used together.

The --summary-display option takes the following values:

no - no summary data will be created. This is the default.

yes - detailed summary data will be created. You can configure the contents with the summary-format option. By default, the following contents are displayed in the summary:

<table>
<thead>
<tr>
<th>Default settings:</th>
<th>Render for example this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source: %c:%g\n&quot;</td>
<td>user@host1#22:/path/to/source/file</td>
</tr>
<tr>
<td>Source parameters: %e\n&quot;</td>
<td>X=TEXT, C=ISO8859-1, D=IBM.1047</td>
</tr>
<tr>
<td>Destination: %C:%G\n&quot;</td>
<td>user@host2#22:/path/to/destination/file</td>
</tr>
<tr>
<td>Destination parameters: %E\n&quot;</td>
<td>NONE</td>
</tr>
<tr>
<td>File size: %s bytes\n&quot;</td>
<td>123456 bytes</td>
</tr>
<tr>
<td>Transferred: %t bytes\n&quot;</td>
<td>123456 bytes</td>
</tr>
<tr>
<td>Rate: %RB/s\n&quot;</td>
<td>345kB/s</td>
</tr>
<tr>
<td>Start: %xy-%xt-%xd %xh:%xm:%xs\n&quot;</td>
<td>2010-01-26 13:10:56</td>
</tr>
<tr>
<td>Stop: %Xy-%Xt-%Xd %Xh:%Xm:%Xs\n&quot;</td>
<td>2010-01-26 13:23:30</td>
</tr>
<tr>
<td>Time: %y \n&quot;</td>
<td>00:12:34</td>
</tr>
</tbody>
</table>

simple - simple one-line summary will be displayed. For example:

```
scpg3 --summary-display=simple sourcefile destinationfile
sourcefile | 127MB | 151.3MiB/s | TOC: 00:00:00 | 100%
```

bytes - basic statistics reporting the transferred bytes will be displayed. For example:

```
scpg3 --summary-display=bytes sourcefile destinationfile
Transferred 12915145984 bytes, file: 'sourcefile' -> 'destinationfile'
```

--summary-format= FORMAT_STRING

Chooses the format and the contents of the summary. You can use this option when --summary-display=yes. Do not use this option with --statistics.

Select the contents for the summary using the following definitions:

```
%c - source connection: user@host#port or profile
%c - destination connection: user@host#port or profile
%D* - current date
%e - source parameters (file transfer and data set parameters)
%E - destination parameters (file transfer and data set parameters)
%f - source file name
%f - destination file name
%g - /path/to/source/file
%g - /path/to/destination/file
%k - compression done ("zlib" or "none")
```
%p - transfer percentage
%q - transfer rate in bit/s
%Q - transfer rate as "XXyb/s" (b/s, kib/s, Mib/s, Gib/s)
%r - transfer rate in bytes/s
%R - transfer rate as "XXyB/s" (B/s, kiB/s, MiB/s, GiB/s)
%s - file size in bytes
%S - file size as "XXyB" (B, kiB, MiB or GiB)
%t - transfer size in bytes
%T - transfer size as "XXyB" (B, kiB, MiB or GiB)
%x* - start date
%x* - end date
%y - elapsed time
%Y - time remaining
%z - ETA or TOC, if transfer has finished
%Z - string "ETA" or "TOC", if transfer has finished

Where * is one of the following:

h - hours (00-23)
m - minutes (00-59)
s - seconds (00-59)
f - milliseconds (0-999)
d - day of the month (1-31)
t - month (1-12)
y - year (1970-)

Other special characters in format strings are:

\n - line feed
\r - carriage return
\t - horizontal tab
\\ - backslash

--progress-display [-no | bar | line ]

Chooses the mode of displaying the progress during a file transfer operation. The default is bar, which shows a progress bar. Option line shows the progress information according to the settings made in the --progress-line-format option.

Do not use this option with --statistics.

--progress-line-format=FORMAT_STRING

Chooses what information will be shown on the progress line. You can use this option when --progress-display=line.

Do not use this option with --statistics.

Select the contents for the progress line using the definitions described for command: --summary-format
--progress-line-interval-seconds

Defines how often the progress information is updated in line mode. The interval is given in seconds, and the default is 60 seconds.

Do not use this option with --statistics.

--streaming [-yes | no | force | ext]

Uses streaming in file transfer, if server supports it. Files smaller than buffer_size_bytes are not transferred using streaming. Use force with small files. Default: yes

Use ext with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The --streaming-ext option requires also the --checksum-no option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

--sunique

Stores data sets with unique names. In case more than one of the transferred data sets have the same name, this feature adds a sequential number to the end of the repeated data set name, for example: DS.version, DS.version1, and DS.version2.

--tcp-connect-timeout=VALUE

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter the timeout value as a positive number. Value 0 (zero) disables this feature and the default system TCP timeout will be used, instead.

--user=USERNAME

Logs in using this user name if the user name is not provided in the address string.

-V, --version

Displays program version and exits.

-h, --help, -?

Displays a short summary of command-line options and exits.

Filename Support

Different operating systems allow different character sets in filenames. On Unix, some of the special characters are allowed in filenames, but on Windows, the following characters are not allowed:

\ / : * ? " < > |

When you use the scp3 command to copy files with special characters (for example unixfilename*?".txt) from a Unix server to Windows, you need to provide the files with new names that are acceptable on Windows. Enter the commands in the following format:
The general rule is to follow your platform specific syntax when you enter filenames containing special characters as arguments to the `scp3` command.

Tectia fully supports filenames containing only ASCII characters. Filenames containing characters from other character sets are not guaranteed to work.

**Using Wildcards**

The `scp3` command supports `*` and `?` as wildcards.

The wildcards can be used both on the remote and the local side in the commands. The following example command will copy all text files (`*.txt`) from all subdirectories of directory `dir2` whose names begin with the prefix `data-` into the current local directory (`.`):

```sh
$ scp3 -r user@server:"dir2/data-*/*.txt" .
```

Note that on Unix, the characters `*` and `?` can appear also in the filenames. So it is necessary to use escape characters to distinguish the wildcards from the characters belonging to a filename. See more information in the section called “Escaping Special Characters”.

**Escaping Special Characters**

Some special characters that are used in filenames in different operating system, may have a special meaning in the Tectia commands. Note also that the meaning can be different in various parts of the file transfer system.

In the `scp3` command, the following characters have a special meaning, and they need to be escaped in commands that take filenames as arguments:

- `*` asterisk is a wildcard for any number of any characters
- `?` question mark is a wildcard for any single character
- `"""` quotation marks placed around strings that are to be taken 'as is'
- `\` backslash is an escape character on Unix
- `~` tilde is an escape character on Windows.

The escape character tells the `scp3` command to treat the next character "as is" and not to assume any special meaning for it. The escape character is selected according to the operating system of the local machine.

Note that the `\` and `~` characters are special characters themselves, and if they are present in the filename, escape characters must be placed in front of them, too. Therefore, if you need to enter a filename containing `\` in Unix or `~` in Windows to the `scp3` command, add the relevant escape character to it:

- `\` on Unix
See the examples below to learn how the escape characters are used in the Tectia `scpg3` command, and how to enter filenames with special characters in different operating systems.

Examples of filenames in the `scpg3` command:

The following filenames are valid in Unix, but they need escape characters in the commands:

```
file|name.txt
file="name".txt
file?name.txt
file*name.txt
file\name.txt
file - name.txt
file~name.txt
```

When using the `scpg3` command on Unix, in certain cases several escape characters are needed, as they escape one another. Enter the above mentioned filenames in the following formats:

```
file\|name.txt     or  "file|name.txt"
file-"name".txt
file\?name.txt   or  "file\?name.txt"
file\*name.txt   or  "file\*name.txt"
file\\name.txt   or  "file\\name.txt"
file\ - \ name.txt  or  "file - name.txt"
file~name.txt
```

Example commands on Unix:

```
$ scpg3  user@server:file\\*name.txt .
$ scpg3  user@server:file\ - \ name.txt .
```

**Environment Variables**

To run `scpg3` on z/OS, the following environment variables must be set:

-_BPXK_AUTOCVT=ON_

If this variable is not set correctly `scpg3` fails to start.

-_BPXK_BATCH_umask=0022_

This variable defines the permissions for newly created files.

-_BPXK_SHAREAS=NO_

This variable defines that `ssh-broker-g3` and `scpg3` processes are run in separate address spaces.

`scpg3` uses the following environment variables:

**SSH_DEBUG_FMT**

This environment variable can be used to specify the format of the debug messages.
For more information, see `SSH_DEBUG_FMT`.

**SSH_SFTP_CHECKSUM_MODE**=yes|no|md5|sha1|md5-force|sha1-force|checkpoint

Defines the setting for comparing checksums. For more information on the available values, see `checksum`.

**SSH_SFTP_HOME_MVS**=yes|no

If this variable is set to `yes`, the `scp3` local directory is set to USER prefix in the MVS side. If it is set to `no` (default), local directory is the current directory.

**SSH_SFTP_SHOW_BYTE_COUNT**=yes|no

If this variable is set to `yes`, the number of transferred bytes is shown after successful file transfer. Also the names of source and destination files are shown. The default is `no`.

**SSH_SFTP_SMF_TYPE**=TYPE119|none

If this variable is set to `TYPE119`, file transfers create SMF records of type 119.

**SSH_SFTP_STATISTICS**=yes|no|simple

If this variable is set to `yes` (default), normal progress bar is shown while transferring the file. If it is set to `no`, progress bar is not shown. If it is set to `simple` file transfer statistics are shown after the file has been transferred.

**SSH_ZCPU_MATH_FACILITY**

Controls the z13 Vector Facility usage. The default value is 1. If this variable is set to 1, `scp3` will probe the availability of the z13 Vector Facility. If the facility is available, `scp3` will use it. If this variable is set to 0 or the facility is not available, `scp3` will not use the z13 vector facility.

**Files**

In addition to the files used by `ssh-broker-g3`, `scp3` uses the following files:

**$HOME/.ssh2/ssh_ftadv_config**

This is the user-specific file that contains a list of file transfer profiles, which furnish file transfer attributes to be used when processing local files. For more information, see Section 9.4.2.

**/opt/tectia/etc/ssh_ftadv_config**

This is the global (system-wide) file that contains a list of file transfer profiles, which furnish file transfer attributes to be used when processing local files. For more information, see Section 9.4.2.

**Exit Values**

`scp3` returns the following values based on the result of the operation:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation was successful.</td>
</tr>
<tr>
<td>1</td>
<td>Internal error.</td>
</tr>
<tr>
<td>2</td>
<td>Connection aborted by the user.</td>
</tr>
<tr>
<td>3</td>
<td>Destination is not a directory, but a directory was specified by the user.</td>
</tr>
</tbody>
</table>
Connecting to the host failed.
Connection lost.
File does not exist.
No permission to access file.
Undetermined error from sshfilexfer.
Some non-fatal errors occurred during a directory operation.
Wrong command-line arguments specified by the user.

Note

When the command is run from JCL using BPXBATCH, the exit values are multiplied by 256.

Examples

Copy files from your local system to a remote Unix system:

```
$ scpg3 localfile user@remotehost:/dst/dir/
```

Copy files from your local system to a remote Windows system:

```
$ scpg3 localfile user@remotehost:/C:/dst/dir/
```

Copy files from a remote system to your local disk:

```
$ scpg3 user@remotehost:/src/dir/srcfile /dst/dir/dstfile
```

Copy files from one remote system to another using connection profiles defined in the ssh-broker-config.xml file:

```
$ scpg3 profile1:/src/dir/srcfile profile2:/dst/dir/dstfile
```
**sftpg3**

sftpg3 -- Secure Shell file transfer client - Generation 3

**Synopsis**

sftpg3 [options...]

[ profile | [user@] host [#port] ]

**Description**

sftpg3 is an FTP-like client that can be used for secure file transfer over the network. sftpg3 launches ssh-broker-g3 to provide a secure transport using the Secure Shell version 2 protocol. ssh-broker-g3 will ask for passwords or passphrases if they are needed for authentication. sftpg3 uses the configuration specified in the ssh-broker-config.xml file.

When started interactively, sftpg3 displays a prompt where the SFTP commands can be entered. It is also possible to start sftpg3 non-interactively with a batch file that contains the commands to be run. For information on the available commands, see the section called “Commands”.

As an alternative to using the command line to set default values for various parameters, it is possible to define the commands in a startup batch file that is run each time sftpg3 is started. By default, sftpg3 looks for a file named ssh_sftp_batch_file in the user-specific directory $HOME/.ssh2/ on Unix or %APPDATA%\SSH\ on Windows.

sftpg3 has two connection end points, local and remote, and both of them can be connected to other hosts than the SFTP client host. If started without arguments, the local end point is connected to the file system of the SFTP client host and the remote end point is unconnected. The connected host(s), with the exception of the SFTP client host, must be running a Secure Shell version 2 server with the sftp-server (or sft-server-g3) subsystem enabled. Tectia Server has sft-server-g3 enabled by default.

The remote connection end point can be given directly as an argument to the sftpg3 command or it can be given with the open SFTP command after sftpg3 has started. The local connection end point can be given with the lopen SFTP command.

When connecting, you can give either the name of a connection profile defined in the ssh-broker-config.xml file (profile) or the IP address or DNS name of the remote host, optionally with the remote user name and the port of the Secure Shell server ([user@] host [#port]). If no user name is given, the local user name is assumed. If no port is given, the default Secure Shell port 22 is assumed.

**Note**

When entering a connection profile in sftpg3, note that Tectia client tools for z/OS deduces the meaning of the argument differently depending on its format. If there is an @ sign in the given at-
tribute value, Tectia client tools for z/OS always interprets it to be `<username@hostname>`, i.e. not a profile.

Also, if there are dots in a profile name (for example `host.x.example.com`, the dots need to be escaped on command line. Enter `host\.x\.example\.com`, instead. Otherwise the profile name is taken as a host name and the current local user name is used for logging in.

For information on special characters in filenames, see the section called “Filename Support”.

**Options**

The following options are available:

- `-4`
  Defines that all connection-related DNS resolutions will be resolved as an IPv4 address.

- `-6`
  Defines that all connection-related DNS resolutions will be resolved as an IPv6 address.

- `-b buffer_size_bytes`
  Defines the maximum buffer size for one SFTP protocol read or write request (default: `32768` bytes).

  The maximum number of SFTP protocol read or write requests sent in parallel within the transfer of a single file can be specified with the `-N` option.

  Note that when streaming (see `--streaming`) is used (as it is by default when transferring files larger than `buffer_size_bytes` to/from Tectia Server), this option is not used for defining buffer sizes.

- `-B [- | batch_file]`
  The `-B` option enables reading from the standard input. This option is useful when you want to launch processes with `sftpg3` and redirect the stdin pipes.

  By defining the name of a `batch_file` as an attribute, you can execute SFTP commands from the given file in batch mode. The file can contain any allowed SFTP commands. For a description of the commands, see the section called “Commands”.

  Using batch mode requires that you have previously saved the server host key on the client and set up a non-interactive method for user authentication (for example, host-based authentication or public-key authentication without a passphrase).

- `-C`
  Disables compression from the current connection.

- `+C`
  Enables zlib compression for this particular connection.
-c, --ciphers=LIST
Sets the allowed ciphers to be offered to the server. List the cipher names in a comma-separated list. For example:

```
--ciphers seed-cbc@ssh.com,aes256-cbc
```

Enter help as the value to view the currently supported cipher names.

-D, --debug=LEVEL
Sets the debug level. LEVEL is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

**Note**

The debug level can be set only when the `sftpg3` command starts the Connection Broker. This option has no effect in the command if the Connection Broker is already running.

-i FILE
Defines that private keys defined in the identification file are used for public-key authentication.

-K, --identity-key-file=FILE
Defines that the given key file of a private key or certificate is used in user authentication. The path to the key file is given in the command.

If the file is a private key, it will be read and compared to the keys already known by the Connection Broker key store. If the key is not known, it will be decoded and added to the key store temporarily. If the file is a certificate and Connection Broker knows a matching private key, it will be used. Both the certificate and the private key can be given using multiple -K options on command line.

-N max_requests
Defines the maximum number of SFTP protocol read or write requests sent in parallel (default: 10).

The size of the buffer used in each read or write request can be specified with the -b option.

Note that this value applies within the transfer of a single file; it cannot be used to define the number of files sent in parallel.

When streaming (see --streaming) is used (as it is by default when transferring files larger than `buffer_size_bytes` specified with the -b option to/from Tectia Server), this option is not used.

-P port
Connects to this Secure Shell port on the remote machine (default: 22).

-q, --quiet
Suppresses the printing of error, warning, and informational messages. This option overrides the quiet-mode setting made in the Connection Broker configuration file.
-v, --verbose
  Uses verbose mode (equal to -D 2).

--aa, --allowed-authentications=METHODS
  Defines the only allowed methods that can be used in user authentication. List the methods in a comma-separated list. For example:

  --allowed-authentications keyboard-interactive,password

  Enter help as the value to view the currently supported authentication methods.

--compressions=METHODS
  Sets the allowed compression methods to be offered to the server. List the methods in a comma-separated list.

  Enter help as the value to view the currently supported compression methods.

--exclusive
  Defines that a new connection will be opened for each connection attempt, otherwise Connection Broker can reuse recently closed connections.

--fips
  Performs the checksums using the FIPS cryptographic library.

--hostkey-algorithms=algorithms
  Sets the allowed host key algorithms to be offered to the server. List the host key algorithms in a comma-separated list. For example:

  --hostkey-algorithms ssh-dss-sha224@ssh.com,ssh-dss-sha256@ssh.com

  Enter help as the value to view the currently supported host key algorithms.

--identity-ID
  Defines that the ID of the private key is used in user authentication. The ID can be Connection Broker-internal ordinary number of the key, the key hash or the key file name.

--identity-key-hash=ID
  Defines the private key used in user authentication with the corresponding public key hash.

--identity-key-id=ID
  Defines that the Connection Broker-internal ordinary number of the key is used in user authentication.

--keep-alive=VALUE
  Defines how often keep-alive messages are sent to the Secure Shell server. Enter the value as seconds. The default value is 0, meaning that keep-alive messages are disabled.
--kexs=KEX

Sets the allowed key exchange (KEX) methods to be offered to the server. List the KEX names in a comma-separated list. For example:

```
--kexs diffie-hellman-group14-sha224@ssh.com,diffie-hellman-group14-sha256@ssh.com
```

Enter `help` as the value to view the currently supported KEX methods.

--kip

Defines keyboard-interactive and password as the allowed methods for user authentication; the same as

```
--allowed-authentications keyboard-interactive,password
```

--macs=LIST

Sets the allowed MACs to be offered to the server. List the MAC names in a comma-separated list. For example:

```
--macs hmac-sha1-96,hmac-md5,hmac-md5-96
```

Enter `help` as the value to view the currently supported MAC names.

--password= PASSWORD | file://PASSWORDFILE | extprog://PROGRAM

Sets the user password or passphrase that the client will send as a response to an authentication method requesting a password or passphrase (hereafter: password). This can be used also with password-protected certificates and public-keys.

The `PASSWORD` can be given directly as an argument to this option (not recommended). Better alternatives are entering a path to a file containing the password (`--password-file://PASSWORDFILE`), or entering a path to a program or script that outputs the password (`--password-extprog://PROGRAM`).

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

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

![Caution]

Supplying the password on the command line is not a secure option. For example, in a multi-user environment, the password given directly on the command line is trivial to recover from the process table. You should set up a more secure way to authenticate. For non-interactive batch jobs, it is more secure to use public-key authentication without a passphrase, or host-based authentication. At a minimum, use a file or a program to supply the password.
--tcp-connect-timeout=VALUE

Defines a timeout period (in seconds) for establishing a TCP connection to the Secure Shell server. Enter a timeout value as a positive number. Value 0 (zero) disables this feature and the default system TCP timeout will be used, instead.

--user=USERNAME

USERNAME will be used in the logon if the user name is not specified in the address string.

-V, --version

Displays program version and exits.

-h, --help, -?

Displays a short summary of command-line options and exits.

Commands

When sftpg3 is ready to accept commands, it will display the prompt sftp>. The user can then enter any of the following commands:

! append ascii auto binary
break cd chmod close continue
debug delete digest echo exit
get getext help ldelete ldigest
lcd lchmod lclose lmkdir locsite
lopen lpwd lreadlink lrename lr
lrm ls lsroots lmdir
lrmdir ls iste lsroots ls
lrmdir ls roots ls
ls symlink lsroots lrmdir
mget mkdir mput open pause
put pwd quit readlink rename
rm rmdir set setext sget
site sput sunique symlink type
verbose

! [command] [arguments]

Invokes an interactive shell on the local machine. If a command is given, it is used as the command to be executed. Optional arguments can be given depending on the command.


Appends the specified local file to the remote file. No globbing can be used.

Options:
-u, --unlink-source

Removes the source file after file transfer.

--streaming [-yes | no | force | ext ]

Uses streaming in file transfer if the server supports it. Files smaller than buffer_size_bytes are not transferred using streaming. Use force with small files. Default: yes

Use ext with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The --streaming=ext option requires also the --checksum-no option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

--force-lower-case

Destination filename will be converted to lowercase characters.

The semantics of options --statistics, --summary-display, --summary-format, --progress-display, --progress-line-format, and --progress-line-interval are the same as with get.

ascii [-s] [remote_nl_conv] [local_nl_conv]

Command ascii sets the transfer mode to ASCII.

For transfers between Tectia on z/OS and other hosts, this also enables automatic ASCII-EBCDIC conversion. Default conversion is between code sets ISO8859-1 and IBM-1047. Files are transferred using the LINE format. The site and lsite commands can be used to change the values.

If you enter the ascii command with any options, it does not set the transfer mode to ASCII, but affects the newline conventions used in the transferred files. You can also set the server's newline convention by using a host profile that specifies the host type. For more information, see the host-type attribute in the section called “The profiles Element”.

Options:

-s

Shows the current newline convention. The line delimiters used in different systems are:

dos: CRLF (\r\n, 0x0d 0x0a)
mac: CR (\r, 0x0d)
mvs: NEL (\n, 0x15)
unix: LF (\n, 0x0a)

remote_nl_conv local_nl_conv

This syntax can be used to define the remote and local newline conventions. The local_nl_conv option operates on the local end, but notice that usually the correct local newline convention is already compiled in.
You can either set hints of the newline conventions for the underlying transfer layer, which by default tries to use the actual newline convention given by the server, or alternatively you can force the newline mode.

To set hints of the newline conventions, use these values in the `remote_nl_conv` and `local_nl_conv` options: dos, unix, and mac. These settings will be used if the remote SSH server does not automatically provide any newline information to the SFTP client. For example:

```
sftp> ascii
File transfer mode is now ascii.
sftp> ascii unix dos
Newline conventions updated.
```

To force the newline conventions, use these values: force-dos, force-unix, and force-mac. These settings force the newline mode irrespective of what the remote SSH server suggests to the SFTP client.

```
sftp> ascii
File transfer mode is now ascii.
sftp> ascii force-unix force-dos
Newline conventions updated.
```

You can also set either one of the options to `ask`, which will cause `sftpg3` to prompt you for the newline convention when needed.

```
auto
File transfer mode will be selected automatically from the file extension.
```

```
binary
Files will be transferred in binary mode.
```

```
brack
Interrupts batch file execution. Batch file execution can be continued with the `continue` command.
```

```
bye
Quits the application.
```

```
cd directory
Changes the current remote working directory.
```

```
chmod [-R] [-f] [-v] OCTAL-MODE [file...]
```

```
chmod [-R] [-f] [-v] [ugoa] [+--] [rwx] [file...]
```

With Unix files, sets file permissions of the specified file or files to the bit pattern `OCTAL-MODE` or changes the file permissions according to the symbolic mode `[ugoa][+-][rwx]`.

Options:
-R
  Recursively changes files and directories.

-\f
  Uses silent mode (error messages are suppressed).

-\v
  Uses verbose mode (lists every file processed).

close
  Closes the remote connection.

continue
  Continues interrupted batch file execution.

debug [ disable | no | debuglevel ]
  Disables or enables debug. With disable or no, debugging is disabled. Otherwise, sets debuglevel as debug level string, as per command-line option -D.

  Tries to delete a file or directory specified in file. The options are the same as for rm.

digest [-H, --hash=] [-o, --offset=] [-l, --length=] file
  Calculates MD5 or SHA-1 digest over file data. The digest is calculated over the data on the disk. If any code or line delimiter conversion attributes are in effect, they are ignored when calculating the digest.

Options:

-\H, --hash= [ md5 | sha1 ]
  Use md5 or sha1 hash algorithm (default: md5).

-\o, --offset=OFFSET
  Start reading from file offset OFFSET.

-\l, --length=LENGTH
  Read LENGTH bytes of file data.

echo Text to be echoed.
  Echo the text. This command can be used for example in batch mode to print text into batch logs.

exit
  Quits the application.

Transfers the specified files from the remote end to the local end. By default, directories are recursively copied with their contents, but this is configurable in the Connection Broker configuration with the SFTP compatibility mode setting (sftpg3-mode in ssh-broker-config.xml). To view the currently set SFTP compatibility mode, run command:

```shell
sftp> help get
```

The currently set compatibility mode is shown in the beginning of the help for command get.

The SFTP compatibility mode options are:

**tectia**

The sftpg3 client transfers files recursively from the current directory and all its subdirectories.

**ftp**

The get command is executed as sget meaning that it transfers a single file, and no subdirectories are copied.

**openssh**

Only regular files and symbolic links from the specified directory are copied, and no subdirectories are copied. Otherwise the semantics of the get command are unchanged.

Options:

- **-p, --preserve-attributes**
  
  Preserves the file permissions and the timestamps when both the source and the destination are on Unix file systems (including z/OS USS). Preserves the timestamps but not the file permissions, if either one, the source or the destination is on Windows. If the destination is on z/OS MVS, none will be preserved.

- **-u, --unlink-source**
  
  Removes the source file after file transfer. Also directories are removed, if they become empty (move mode).

- **-I, --interactive**
  
  Prompts whether to overwrite an existing destination file (does not work with batch mode).

- **--overwrite [ -yes | no ]**
  
  Decides whether to overwrite existing destination file(s) (default: yes).

- **--checksum [ -yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]**
  
  Uses MD5 or SHA-1 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than buffer_size_bytes are not checked. Use...
md5-force or sha1-force with small files (default: yes, i.e. use MD5 checksums). Use checkpointing when transferring large files one by one.

--whole-file

Does not try incremental checks. By default (if this option is not given), incremental checks are tried. This option can only be used together with the --checksum option.

--checkpoint-s<seconds>

Time interval between checkpoint updates (default: 10 seconds). This option can only be used when --checksum=checkpoint.

--checkpoint-b<bytes>

Byte interval between checkpoint updates (default: 10 MB). This option can only be used when --checksum=checkpoint.

--streaming [yes | no | force | ext ]

Uses streaming in file transfer if the server supports it. Files smaller than buffer_size_bytes are not transferred using streaming. Use force with small files. Default: yes

Use ext with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The --streaming=ext option requires also the --checksum=no option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

An alternative way to activate extended streaming is to define SSH_SFTP_STREAMING_MODE=ext and SSH_SFTP_CHECKSUM_MODE=no as environment variables.

--force-lower-case

Destination file name will be converted to lower case characters.

--max-depth=VALUE

Defines whether directories are copied recursively. The value can be:

0 - unlimited recursion, directories are recursively copied with their contents

1 - copies files from the specified directory only, not from subdirectories

2-n - copies files recursively from the specified number of directory levels. Here n means the system-specific maximum.

This command line option overrides the recursion depth set in the Connection Broker configuration with element sftpg3-mode and/or the setting made using environment variable SSH_SFTP_CMD_GETPUT_MODE.
--prefix=PREFIX

Adds prefix PREFIX to filename during the file transfer. The prefix is removed after the file has been successfully transferred.

On z/OS, when applied to MVS data set names, the prefix will be inserted after the High Level Qualifier (HLQ) by default. In case you want the prefix to be a separate qualifier, include a dot at the end of the prefix:

--prefix=PREFIX.

--statistics [ -no | yes | simple ]

**Note**

In release 6.1.5, the behavior of the --statistics option has changed and the --statistics-format option has been removed. Instead of them, use the new --summary-display and --summary-format options.

The --statistics option chooses the style of the statistics to be shown after a file transfer operation. Note that --statistics and --summary-display must not be used together.

The --statistics option takes the following values:

no - no statistics will be created.

yes - shows a progress bar during the file transfer. This is the default. An example of the output:

```
sftp> get --statistics="yes" sourcefile
sourcefile               |  127MB  |  42.9MiB/s | TOC: 00:00:03 | 100%
```

simple - simple one-line statistics will be displayed after the file transfer has ended. For example:

```
sftp> get --statistics=simple testfile
sourcefile   | 127MB  | 151.3MiB/s | TOC: 00:00:00 | 100%
```

--summary-display [ -no | yes | simple | bytes ]

Chooses the style of the file transfer summary data to be displayed after a file transfer operation. With the summary display, the progress bar data is also displayed by default.

Note that --summary-display and --statistics must not be used together.

The --summary-display option takes the following values:

no - no summary data will be created. This is the default.

yes - detailed summary data will be created. You can configure the contents with the summary-format option. By default, the following contents are displayed in the summary:

```
Default settings:  Render for example this:
"Source: %c:%g\n"        user@host1#22:/path/to/source/file
```
simple - simple one-line summary will be displayed. For example:

```bash
sftp> get --summary-display=simple sourcefile
sourcefile   | 127MB   | 151.3MiB/s | TOC: 00:00:00 | 100%
```

bytes - basic statistics reporting the transferred bytes will be displayed. For example:

```bash
sftp> get --summary-display=bytes sourcefile
Transferred 12915145984 bytes, file: 'sourcefile' -> 'destinationfile'
```

```
--summary-format=FORMAT_STRING

Chooses the format and the contents of the summary. You can use this option when --summary-display=yes. Do not use this option with --statistics.
```

Select the contents for the summary using the following definitions:

```bash
%c  - source connection: user@host#port or profile
%e  - source parameters (file transfer and data set parameters)
%f  - source file name
%g  - /path/to/source/file
%k  - compression done ("zlib" or "none")
%p  - transfer percentage
%q  - transfer rate in bit/s
%r  - transfer rate in bytes/s
%S  - file size as "XXyB" (B, kiB, MiB or GiB)
%t  - file size in bytes
%T  - transfer size as "XXyB" (B, kiB, MiB or GiB)
%y  - elapsed time
%X  - end date
```
Where * is one of the following:

h = hours (00-23)
m = minutes (00-59)
s = seconds (00-59)
f = milliseconds (0-999)
d = day of the month (1-31)
t = month (1-12)
y = year (1970-)

Other special characters in format strings are:

\n = line feed
\r = carriage return
\t = horizontal tab
\\ = backslash

--progress-display [ -no ] [ bar ] [ line ]
Chooses the mode of displaying the progress during a file transfer operation. The default is bar, which shows a progress bar. Option line shows the progress information according to the settings made in the --progress-line-format option.

Do not use this option with --statistics.

--progress-line-format=FORMAT_STRING
Chooses what information will be shown on the progress line. You can use this option when --progress-display=line.

Do not use this option with --statistics.

Select the contents for the progress line using the definitions described for option --summary-format of the get command above.

--progress-line-interval=seconds
Defines how often the progress information is updated in the line mode. The interval is given in seconds, and the default is 60 seconds.

Do not use this option with --statistics.

getext
Displays the extensions that will be ASCII in the auto transfer mode.

lappend [ options... ] srcfile [ dstfile ]
The same as append, but appends the specified remote file to the local file.

lcd directory
Changes the current local working directory.
1chmod [-R] [-f] [-v] OCTAL-MODE [file...]

1chmod [-R] [-f] [-v] [ugoa] [+--] [rwx] [file...]

The same as chmod, but operates on local files.

1close

Closes the local connection.

1delete [options...] file...

The same as delete, but operates on local files.

1digest [-H, --hash] [-o, --offset] [-l, --length] file

The same as digest, but operates on local files.

1ls [-R] [-l] [-S] [-r] [-p] [-z|+z] [file...]

The same as ls, but operates on local files.

1lsroots

The same as lsroots, but operates on local files (when the local end has been opened to a VShell server).

1mkdir directory

The same as mkdir, but operates on local files.

localopen [user@hostname[#port][-l][-user=USERNAME]]

The same as lopen.

1open [user@hostname[#port][-l][-user=USERNAME]]

Tries to connect the local end to the host hostname. If this is successful, lls and friends will operate on the file system on that host.

Options:

-1

Connects the local end to the file system of the SFTP client host (which does not require a server). This is also the default state when no lopen commands have been given.

--user

Defines the user in the connection to be USERNAME.

locsite [none| name1=value1 name2=value2... ]

The same as site, but operates on local files and data sets.

1pwd

Prints the name of the current local working directory.

1readlink path

The same as readlink, but operates on local files.
rename  oldfile newfile
    The same as rename, but operates on local files.

lrm [options...] file...
    The same as rm, but operates on local files.

lrmdir directory
    The same as rmdir, but operates on local files.

ls [ -R ] [ -l ] [ -S ] [ -r ] [ -p ] [ -z | +z ] [ file... ]
    Lists the names of files on the remote server. For directories, contents are listed. If no arguments are
given, the contents of the current working directory are listed.

    Options:
    
    -R
    Directory trees are listed recursively. By default, subdirectories of the arguments are not visited.

    -l
    Permissions, owners, sizes and modification times are also shown (long format).

    -S
    Sorting is done based on file sizes (default: alphabetical sorting).

    -r
    The sort order is reversed.

    -p
    Only one page of listing is shown at a time.

    -z
    The client generates the long output.

    +z
    The long output supplied by the server is used, if available (alias for option -l).

lsite [ none | name1=value1 name2=value2... ]
    The same as site, but operates on local files and data sets.

lsroots
    Dumps the virtual roots of the server. (This is a VShell extension. Without this you cannot know the file
system structure of a VShell server.)

lsymlink targetpath linkpath
    The same as symlink, but operates on local files.

mget [options...] file...
    Synonymous to get.
mkdir directory
    Tries to create the directory specified in directory.

mput [options...] file...
    Transfers the specified files from the local end to the remote end. Options and semantics are the same as for get. Synonymous to put.

open [user@]hostname[#port][-l][--user=USERNAME]
    Tries to connect the remote end to the host hostname.

    Options:

    -l
        Connects the remote end to the file system of the SFTP client host (which does not require a server).

    --user
        Defines the user in the connection to be USERNAME.

pause [seconds]
    Pauses batch file execution for seconds seconds, or if seconds is not given until ENTER is pressed.

put [options...] file...
    Transfers the specified files from the local end to the remote end. Options and semantics are the same as for get.

pwd
    Prints the name of the current remote working directory.

quit
    Quits the application.

readlink path
    Provided that path is a symbolic link, shows where the link is pointing to.

rename oldfile newfile
    Tries to rename the oldfile to newfile. If newfile already exists, the files are left intact.

rm [-I, --interactive][-r, --recursive] file...
    Tries to delete a file or directory specified in file.

    Options:

    -I, --interactive
        Prompts whether to remove a file or directory (does not work with batch mode).

    -r, --recursive
        Directories are removed recursively.
rmdir directory
Tries to delete the directory specified in directory. This command removes the directory only if it is empty and has no subdirectories.

set [defaults| [--commands=name1,name2,... exit-value=VALUE] | option1=value1 option2=value2...
]  
Sets the default values for various parameters. The set command takes the following options:

defaults
Sets the parameters to be system defaults.

checksum [-yes | no | md5 | sha1 | md5-force | sha1-force | checkpoint ]
Uses MD5 or SHA-1 checksums or a separate checkpoint database to determine the point in the file where file transfer can be resumed. Files smaller than buffer_size_bytes are not checked. Use md5-force or sha1-force with small files. The default is md5 (in z/OS the default is no). Use checkpointing when transferring large files one by one.

compatibility-mode [-tectia | ftp | openssh ]
Defines what mode of recursiveness is used in the file transfer:

tectia
The sftp3 client transfers files recursively from the current directory and all its subdirectories. This is the default mode.

ftp
A single file is transferred, and no subdirectories are copied.

openssh
Only regular files and symbolic links from the specified directory are copied, and no subdirectories are copied.

compressions [-none | zlib ]
Defines whether compression is used in file transfer:

none
Compression is not used. This is the default.

zlib
Enables zlib compression in file transfer.

exit-value=VALUE
Defines the exit value of sftp3 in batch mode in case of an error. The value must be between 0 and 255. If exit-value is set to something else than 0 and the --commands parameter is not used, batch execution terminates when the first error occurs.

Example 1: If the rename command in this batch job fails, sftp3 will stop and return exit value "6":

Example 2: If you want to ignore a possible failure of a specific command and return exit value "0" independent of the actual result of the operation, use `set exit-value=0` after the command. This example batch job ignores possible failure in renaming a file:

```
open user@host
rename file file2
set exit-value=0
<next command in batch job>
quit
```

```
--commands=name1,name2,... exit-value=VALUE
```

This option makes an `sftp3` batch job abort when any of the specified commands fail. When a command that is not specified with this option fails, the batch job execution continues, and the exit value of the batch job is set to the one defined with `exit-value`. Note that if `exit-value=0`, the exit value of the failed command will be returned.

Example 3: When `sftp3` is running in batch mode, it will abort execution if a `put`, `get`, or `ls` command fails. If any other command (with the exceptions mentioned below) fails, execution will continue until the end of the batch file. In both cases value "3" will be returned:

```
set --commands=put,get,ls exit-value=3
```

Example 4: When `sftp3` is running in batch mode, it will abort execution when a `put` or `get` command fails. If any other command (with the exceptions mentioned below) fails, execution will continue. In any case the original exit value of the last failed command will be returned:

```
set --commands=put,get exit-value=0
```

Exceptions: When `exit-value` is set for specific commands with the `--commands` option, also the following situations will cause the batch job execution to abort:

- A `cd` command resulting in an error
- Any invalid command
- Authentication failed error
- Unable to connect to server error
- Connection aborted error

By default (`set defaults`), in case of errors, `sftp3` does not stop but instead will continue executing and return the last error message.
Invalid commands added in --commands will be ignored.

overwrite [ -yes | no ]

Decides whether to overwrite existing destination file(s) (default: yes).

progress-display [ -bar | line | no ]

Chooses the mode of displaying the progress during a file transfer operation. The default is bar, which shows a progress bar. Option line shows the progress information according to the settings made with the progress-line-format option. Option no disables progress display.

progress-line-format=FORMAT_STRING

Chooses what information will be shown on the progress line. Use this option when --progress-display=line. See the definitions of contents options in command: get --progress-line-format.

progress-line-interval=seconds

Defines how often the progress information is updated in line mode. The interval is given in seconds, and the default is 60 seconds.

summary-display [ -no | yes | simple | bytes ]

Chooses the style of the file transfer summary data to be displayed after a file transfer operation. With the summary display, the progress bar data is also displayed by default. Do not use this option with --statistics.

See the options described for command: get --summary-display

summary-format=FORMAT_STRING

Chooses the format and the contents of the summary. You can use this option when --summary-display=yes. Do not use this option with --statistics.

See the definitions of contents options in command: get --summary-format

streaming [ -yes | no | force | ext ]

Uses streaming in file transfer if the server supports it. Files smaller than buffer_size_bytes are not transferred using streaming. Use force with small files. Default: yes.

Use ext with z/OS hosts to enable direct MVS data set access. Use this option only when the file transfer is mainly used for mainframe data set transfers, as it can slow down the transfer of small files in other environments.

The streaming=ext option requires also the checksum=no option, because if checksums are calculated, the file transfer uses staging, which excludes streaming.

setext [extension...]

Sets the file extensions that will be ASCII in the auto transfer mode. Normal zsh-fileglob regexps can be used in the file extensions.
sget [options...] srcfile [dstfile]

Transfers a single specified file from the remote end to the local end under the filename defined with dstfile. Directories are not copied. No wildcards can be used. Options are the same as for get.

site [ none | name1=value1 name2=value2... ]

Sets the file and data set parameters for the remote host. Parameters can be entered either one by one, or several parameters can be delimited by spaces or commas. Both long parameters and abbreviations can be used. When run without arguments, the site command outputs the list of entered parameters. Setting none resets all parameters.

The available parameters are:

- AUTOMOUNT=YES|NO|IMMED
- [NO]AUTOMOUNT|[NO]AUTOM
- AUTORECALL=YES|NO
- [NO]AUTORECALL|[NO]AUTOR
- BLKSIZE|B|BLOCKSI-size
- BLOCKS|BL
- CONDDISP|CO=CATLG|UNCATLG|KEEP|DELETE
- CYLINDERS|CY
- DATACLAS|DA=class
- DATASET_SEQUENCE_NUMBER|SEQNUM=number
- DEFER|DE=YES|NO
- [NO]DEFER|DE
- DIRECTORY_SIZE|M|DI|DIRSZ-size
- EXPIRY_DATE|EXPDT=yyddd|yyyyddd
- FILE_STATUS|STATUS=NEW|MOD|SHR|OLD
- FILETYPE|FILET=SEQ|JES
- FIXRECFM|FI=length
- JOB_ID|JESID=ID
- JOB_OWNER|JESO=name
• JOBNAME | JESJOB=name
• KEYLEN | KEYL=length
• KEYOFF | KEYO=offset
• LABEL_TYPE | LABEL=NL|SL|NSL|SUL|BLP|LTM|AL|AUL
• LIKE=like
• LRECL | R | LR=length
• MGMTCLAS | MG=class
• NORMDISP | NOR=CATLG|UNCATLG|KEEP|DELETE
• PRIMARY_SPACE | PRI=space
• PROFILE | P | PROF=profile
• RECFM | O | REC=recfm
• RECORD_TRUNCATE | U | TRUN=YES|NO
• [NO]TRUNCATE | [NO]TRU | [NO]TRUN
• RETENTION_PERIOD | RET=days
• SECONDARY_SPACE | SE | SEC=space
• SIZE | L=size
• SPACE_RELEASE | RLSE=YES|NO
• SPACE_UNIT | SU=BLKS|TRKS|CYLS|AVGRECLEN
• SPACE_UNIT_LENGTH | SUL=length
• STAGING | S | STAGE=YES|NO
• STORCLAS | ST=class
• SVC99_TEXT_UNITS | SVC99=string
• TRACKS | TR
• TRAILING_BLANKS | TRAIL=YES|NO
• [NO]TRAILINGBLANKS | [NO]TRAI | [NO]TRAIL
• TRANSFER_CODESET | C | CODESET=codeset
• TRANSFER_FILE_CODESET|D|FCODESET=codeSet
• TRANSFER_FILE_LINE_DELIMITER|J|FLDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL
• TRANSFER_FORMAT|F|FORMAT=LINE|STREAM|RECORD
• TRANSFER_LINE_DELIMITER|I|LDELIM=UNIX|MVS|MVS-FTP|DOS|MAC|NEL
• TRANSFER_MODE|X|MODE=BIN|TEXT
• TRANSFER_TRANSLATE_DSN_TEMPLATES|A|XDSNT=templates
• TRANSFER_TRANSLATE_TABLE|E|XTBL=table
• TYPE|T=PS|PO|PDS|POE|PDSE|GDG|HFS|VSAM|ESDS|KSDS|RN
• UNIT|UN=unit
• UNIT_COUNT|UC|UNC=number
• UNIT_PARALLEL|UNP=YES|NO
• VOLUME_COUNT|VC|VOLCNT=number
• VOLUMES|VO|VOL=vol1+vol2+

For a detailed description of the parameters, see Section 6.4.1.

sput [options...] srcfile [dstfile]
  Transfers a single specified file from the local end to the remote end under the filename defined with dstfile. Directories are not copied. No wildcards can be used. Options are the same as for get.

sununique [on] [off]
  Stores files with unique names. If no option is specified, the command toggles the state of 'sununique'.

  In case more than one of the transferred data sets have the same name, this feature adds a sequential number to the end of the repeated data set name, for example: DS.version, DS.version1, and DS.version2.

symlink targetpath linkpath
  Creates symbolic link linkpath, which will point to targetpath.

type [ascii | auto | binary | default ]
  Sets file transfer type. If type is not specified, the current file transfer type is displayed.

  ascii
    Transfer file in ascii mode. See ascii for more information.
auto
Transfer file in auto mode. See auto for more information.

binary
Transfer file in binary mode. See binary for more information.

default
Transfer file in binary mode. This mode is identical to binary except that when the server is Tectia Server for IBM z/OS, no extra parameters are specified for the server. See binary for more information.

verbose
Enables verbose mode (identical to the debug 2 command). You may later disable verbose mode by debug disable.

help [topic]
If topic is not given, lists the available topics. If topic is given, outputs available online help about the topic.

helpall
Outputs available online help about all topics.

Command Interpretation

sftpg3 understands both backslashes (\) and quotation marks ("") on the command line. A backslash can be used for ignoring the special meaning of any character in the command-line interpretation. It will be removed even if the character it precedes has no special meaning.

When specifying filenames that contain spaces, enclose them in quotation marks.

Note

Commands get and put will get or put every file in the current directory and possibly overwrite files in your current directory.

sftpg3 supports wildcard characters (also known as glob patterns) given to commands chmod, lchmod, ls, lls, rm, lrm, get, and put.

Command-Line Editing

The following key sequences can be used for command-line editing:

Ctrl-Space
Set mark.
**Ctrl-A**
Go to the beginning of the line.

**Ctrl-B**
Move the cursor one character to the left.

**Ctrl-D**
Erase the character to the right of the cursor, or exit the program if the command line is empty.

**Ctrl-E**
Go to the end of the line.

**Ctrl-F**
Move the cursor one character to the right.

**Ctrl-H**
Backspace.

**Ctrl-I**
Tab.

**Ctrl-J**
Enter.

**Ctrl-K**
Delete the rest of the line.

**Ctrl-L**
Redraw the line.

**Ctrl-M**
Enter.

**Ctrl-N**
Move to the next line.

**Ctrl-P**
Move to the previous line.

**Ctrl-T**
Toggle two characters.

**Ctrl-U**
Delete the line.

**Ctrl-W**
Delete a region (the region's other end is marked with Ctrl-Space).
Ctrl-X
  Begin an extended command.

Ctrl-Y
  Yank deleted line.

Ctrl-_
  Undo.

Ctrl-X Ctrl-L
  Lower case region.

Ctrl-X Ctrl-U
  Upper case region.

Ctrl-X Ctrl-X
  Exchange cursor and mark.

Ctrl-X H
  Mark the whole buffer.

Ctrl-X U
  Undo.

Esc Ctrl-H
  Backwards word delete.

Esc Delete
  Backwards word delete.

Esc Space
  Delete extra spaces (leaves only one space).

Esc <
  Go to the beginning of the line.

Esc >
  Go to the end of the line.

Esc @
  Mark current word.

Esc A
  Go back one sentence.

Esc B
  Go back one word.
Esc C
    Capitalize current word.

Esc D
    Delete current word.

Esc E
    Go forward one sentence.

Esc F
    Go forward one word.

Esc K
    Delete current sentence.

Esc L
    Change current word to lower case.

Esc T
    Transpose words.

Esc U
    Change current word to upper case.

Delete
    Backspace.

**Filename Support**

Different operating systems allow different character sets in filenames. On Unix, some of the special characters are allowed in filenames, but on Windows, the following characters are not allowed:

\/:*?"<>|  

The `sftpg3` command-line tool (both as an interactive and in a batch file) follows the syntax and semantics of Unix shell command-line also on the Windows platform, except that the escape character is ~ (tilde).

When you transfer files that have special characters in the filename (for example `unixfilename*?".txt`) from a Unix server to Windows, you need to provide the files with new names that are acceptable on Windows.

The `sftpg3` command-line client includes two versions of the `get` command:

The `get` command can be used to transfer several files at the same time, but it is not possible to define target filenames. Note that if there are special characters in the filenames, you need to rename the files already on Unix so that the names are acceptable also on Windows.

The `sget` command is used to transfer one file at a time, and it allows you to define a new name for the destination file. Use it to make the name acceptable on Windows. The command sequence is as follows:
Escaping special characters

In the `sftpg3` command, the following characters have a special meaning, and they need to be escaped in commands that take filenames as arguments:

* asterisk is a wildcard character for any number of any characters

? question mark is a wildcard for any single character

""" quotation marks are placed around strings that are to be taken 'as is'

\backslash is an escape character on Unix

~ tilde is an escape character on Windows

The escape character tells the `sftpg3` command to treat the next character "as is" and not to assume any special meaning for it. The escape character is selected according to the operating system of the local machine.

Note that the \ and ~ characters are special characters themselves, and if they are present in the filename, an escape character must be placed in front of them, too. Therefore, if you need to enter a filename containing \ in Unix or ~ in Windows to any of the `sftpg3` commands, add the relevant escape character to it:

\ on Unix

~~ on Windows

When a filename or part of a filename is placed within the quotation marks "", the `sftpg3` command interprets the quoted part 'as is', and none of the characters within the quote are interpreted as wildcards or as any other special characters.

However, on Unix a quotation mark '"' can also be part of a filename. If you need to enter the '"' character in a filename, you must add the escape character in front of it both on Unix and on Windows.

For example, to enter a file named `file-"name".txt` into a command on Windows, enter the following command:

`sftp> sget "file-"name".txt" filename.txt`

See the examples below to learn how the escape characters are used in the Tectia `sftpg3` commands, and how to enter filenames with special characters in different operating systems.

Examples of filenames in the `sftpg3` commands:

The following filenames are valid in Unix, but they need escape characters in the commands:
When using the sftpg3 command-line tool on Unix, enter the above mentioned filenames in the following formats:

```
file\name.txt or "file\name.txt"
file-"name".txt or "file-"name".txt"
file?name.txt or "file?name.txt"
file*name.txt or "file*name.txt"
file\name.txt or "file\name.txt"
file - name.txt or "file - name.txt"
file~name.txt or "file~name.txt"
```

Example commands on Unix:

```
sftp> get "file*name.txt"
sftp> sget "file*name.txt" newfilename.txt
```

**Environment Variables**

In order to run sftpg3 the following environment variables must be set:

**_BPXK_AUTOCVT_**=ON

If this variable is not set correctly sftpg3 fails to start.

**_BPXK_BATCH_UMASK_**=0022

This variable defines the permissions for newly created files, and this umask value will be used if the server configuration file does not have a umask defined for the sft-server-g3 subsystem.

**_BPXK_SHAREAS_**=NO

This variable defines that ssh-broker-g3 and sftpg3 processes are run in separate address spaces.

sftpg3 uses the following environment variables:

**SSH_DEBUG_FMT**

This environment variable can be used to specify 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"
```

It will produce an output similar to the following example:

```
```
The format of the debug message is composed of flags that are percent signs (%) followed by one or more characters. If no % is used, the character will be literally printed to the debug message. The value of the flags will be shown only if the information is available.

The different flags that can be used are:

```
%u
   Prints the uid.
%e
   Prints the euid.
%p
   Prints the pid.
%t
   Prints the thread id.
%h
   Prints the hostname.
%l
   Prints the debug level.
%m
   Prints the module.
%n
   Prints the line.
%f
   Prints the function name.
%s
   Prints the filename.
%M
   Prints the message.
%o
   Prints the ordinal or the number of messages printed so far.
%N
   Prints the new line.
%E(var)
   Expands to the value of the environment variable var.
```
%Dx

Formats a piece of the current date, where x is one of the following

- f -- milliseconds
- s -- seconds
- m -- minutes
- h -- hours
- d -- day
- t -- month
- y -- year

%Rx

Reports resource usage, where x is one of the following

- u -- user time used by current process
- U -- user time used by children that have terminated and have been waited for
- s -- system time used by current process
- S -- system time used by children that have terminated and have been waited for

%W(m) (i)

Enables word-wrapping of debug messages. No line of output will be longer than m characters, and every line after the first one in a message will be indented by i spaces. This does not actually output anything.

%c(n)

Just writes the character n in verbatim to the output, and it is assumed to take zero width.

%s(n)

Causes the debug module to sleep n milliseconds after the debug message has been printed. The value of n must lie between 0 and 60,000 (one minute).

%<(n)x

Gives field maximum width n to the option x.

%>(n)x

Gives field minimum width n to the option x.

%$x

Specifies alignment to right.

**SSH_SFTP_BATCH_FILE=startup_batch_file**

Defines the path to the sftp3 startup batch file. The file is run and the sftp3 commands defined in the file are executed each time sftp3 is started.

If this variable is not defined, sftp3 looks for a startup batch file named ssh_sftp_batch_file in the user-specific directory $HOME/.ssh2/ on Unix or %APPDATA%\SSH\ on Windows.

Note that if this variable is defined but the file is missing or cannot be accessed, sftp3 fails to start.
**SSH_SFTP_CHECKSUM_MODE**—yes|no|md5|sha1|md5-force|sha1-force|checkpoint

Defines the setting for comparing checksums. For more information on the available values, see checksum.

**SSH_SFTP_HOME_MVS**—yes|no

If this variable is set to yes, the sftpg3 local directory is set to USER prefix in the MVS side. If it is set to no (default), local directory is the current directory.

**SSH_SFTP_SHOW_BYTE_COUNT**—yes|no

If this variable is set to yes, the number of transferred bytes is shown after successful file transfer. Also the names of source and destination files are shown. The default is no.

**SSH_SFTP_SMF_TYPE**—TYPE119|none

If this variable is set to TYPE119, file transfers create SMF records of type 119.

**SSH_SFTP_STATISTICS**—yes|no|simple

If this variable is set to yes (default), normal progress bar is shown while transferring the file. If it is set to no, progress bar is not shown. If it is set to simple, file transfer statistics are shown after the file has been transferred.

**SSH_ZCPU_MATH_FACILITY**

Controls the z13 Vector Facility usage. The default value is 1. If this variable is set to 1, sftpg3 will probe the availability of the z13 Vector Facility. If the facility is available, sftpg3 will use it. If this variable is set to 0 or the facility is not available, sftpg3 will not use the z13 vector facility.

**Files**

In addition to the files used by ssh-broker-g3, sftpg3 uses the following files:

$HOME/.ssh2/ssh_ftadv_config

This is the user-specific file that contains a list of file transfer profiles, which furnish file transfer attributes to be used when processing local files. For more information, see Section 9.4.2.

/opt/tectia/etc/ssh_ftadv_config

This is the global (system-wide) file that contains a list of file transfer profiles, which furnish file transfer attributes to be used when processing local files. For more information, see Section 9.4.2.

**Exit Values**

sftpg3 returns the following values based on the result of the operation:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation was successful.</td>
</tr>
<tr>
<td>1</td>
<td>Internal error.</td>
</tr>
<tr>
<td>2</td>
<td>Connection aborted by the user.</td>
</tr>
<tr>
<td>3</td>
<td>Destination is not a directory, but a directory was specified by the user.</td>
</tr>
<tr>
<td>4</td>
<td>Connecting to the host failed.</td>
</tr>
<tr>
<td>5</td>
<td>Connection lost.</td>
</tr>
</tbody>
</table>
6     File does not exist.
7     No permission to access file.
8     Undetermined error from sshfilexfer.
11    Some non-fatal errors occurred during a directory operation.
101   Wrong command-line arguments specified by the user.

**Note**

When the command is run from JCL using BPXBATCH, the exit values are multiplied by 256.

In batch mode, *sftpg3* returns the value 0 only if no errors occurred during the execution. A failure to change the current working directory, a failure to establish a connection, or a connection loss during batch operation cause *sftpg3* to abort. Other errors are reported to stderr and the last error value is returned as the exit value of the *sftpg3* process.

**Note**

When the command is run from JCL using BPXBATCH, the exit values are multiplied by 256.

**Examples**

Open a *sftpg3* session with the remote end connected to the server defined in the connection profile `profile1` in the `ssh-broker-config.xml` file (the local end is initially connected to the file system of the SFTP client host):

```bash
$ sftpg3 profile1
```

Run *sftpg3* in batch mode:

```bash
$ sftpg3 -B batch.txt
```

Example contents of the batch file `batch.txt` are shown below. Non-interactive authentication methods are used and the server host keys have been stored beforehand:

```bash
lopen user@unixserver.example.com
open user@winserver.example.com
binary
lcd backup
cd c:/temp
get --force-lower-case Testfile-X.bin
lchmod 700 testfile-x.bin
quit
```

The example batch file opens the local end of the connection to a Unix server and the remote end to a Windows server, and sets the transfer mode to binary. It changes to local directory `backup` and remote directory `C:\Temp`, and copies a file from the remote directory to the local directory. The filename is changed to lower-case characters (`testfile-x.bin`). After transfer, the file permissions are changed to allow the user full rights and others no rights.
ssh-translation-table

ssh-translation-table -- Secure Shell Translation Table

Synopsis

ssh-translation-table [options...] [filename]

Description

ssh-translation-table is a utility program that generates translation tables for coded character set (CCS) conversions. ssh-translation-table stores the translation table in filename. If filename is not given, ssh-translation-table writes the translation table to standard output.

Options

The following options are available:

- `--binary`  
  Use the z/OS-specific binary file format.

- `--from CODESET`
  Specify the source code set of the inbound conversion, which is also the target code set of the outbound conversion. The default value is `ISO8859-1`. For example:

  ```
  --from ISO8859-15
  ```

- `--to CODESET`
  Specify the target code set of the inbound conversion, which is also the source code set of the outbound conversion. The default value is `IBM-1047, swapifnl` if the underlying implementation is ICU, otherwise `IBM-1047`. For example:

  ```
  --to IBM-037
  ```

- `--list-charsets`
  List available character sets. Note that all character sets are not single byte character sets. Only single byte character sets can be used.

- `--debug LEVEL`
  Sets the debug level. LEVEL is a number from 0 to 99, where 99 specifies that all debug information should be displayed. This should be the first argument on the command line.

- `--help`
  Displays a short summary of command-line options and exits.
Translation Table

A translation table is a file containing two tables describing the character conversion, the inbound table and the outbound table. Each table consists of 256 target values.

In Tectia File Transfer, the inbound table is used when converting data from the line to the data set. The outbound table is used when converting data from a file and sending the data out on the line.

The binary format, which is z/OS specific, consists of three 256 byte fields. The first is a comment in EBCDIC, which is ignored in the conversion software, the second is the inbound table and the third is the outbound table.

The text format can have interspersed comments. The target values are in hexadecimal.

A table is a list of 256 values represented as two hexadecimal characters (from 00 to FF). The position of the value is the index for conversion. The first position, i.e. position 00, represents the converted value for byte value of 0.

The hexadecimal values in the tables are case-insensitive. So values 0a and 0A are the same. Also, it is possible to add comments into the file. The comment starts with character '#'. Everything after that until end of line is treated as comment and ignored. Also all white spaces are ignored.

Note

Only single byte translations are supported with translation tables.

Here is an example translation table generated with command ssh-translation-table:

```plaintext
## SSH TRANSLATION TABLE FILE FORMAT VERSION 1.0
###########################################################
#
# This file is an example translation table that can be used to
# translate data from 'ISO8859-1' to 'IBM-1047,swaplfnl' while reading
# from a file or from 'IBM-1047,swaplfnl' to 'ISO8859-1' while writing
# to a file.
#
# The format of translation table file is following:
#
# - White spaces are ignored.
# - Everything after '#' character until end of line is a comment
#   that is ignored.
# - The first table is used when writing data to a file.
# - The second table is used when reading data from a file.
# - Both tables must exist.
# - Table is a simple hexadecimal representation of the
#   translation. Each value is represented as two hexadecimal
#   characters. The first line gives the values in table
#   positions 0-15 (00-0F), the second line 16-31 (10-1F)
# and so on.
```
# Note: Only single byte translations are supported.
#
### Inbound (network to file) translation table:
# IBM-1047,swaplfnl -> ISO8859-1

<table>
<thead>
<tr>
<th>0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>000102039C09867F978D8E0B0C0D0E0F #0</td>
</tr>
<tr>
<td>1011112139D0A08871819928F1C1D1E1F #1</td>
</tr>
<tr>
<td>808182838485171B88898A8B8C050607 #2</td>
</tr>
<tr>
<td>909116939495960498999A9B14159E1A #3</td>
</tr>
<tr>
<td>20A0E2E4E0E1E3E5E7F1A2E3C282B7C #4</td>
</tr>
<tr>
<td>26E9EABEB8EDEEEFECDF21242A293B5E #5</td>
</tr>
<tr>
<td>2D2FC2C4C0C1C3C5C7D1A62C255F3E3F #6</td>
</tr>
<tr>
<td>F8C9CABBC8CDCEFC603A2340273D22 #7</td>
</tr>
<tr>
<td>D8612623646566676869ABBF0DFEB1 #8</td>
</tr>
<tr>
<td>B06A6B6C6D6E6F707172AABAE6B8C6A4 #9</td>
</tr>
<tr>
<td>B57E737475767778797A1BF05BDEAE A</td>
</tr>
<tr>
<td>ACA3A5B7A9A7B6BCBDEDADA8AF5DB4D7 B</td>
</tr>
<tr>
<td>7B4142434445464748494DF4F6F2F3F5 C</td>
</tr>
<tr>
<td>7D4A4B4C4D4E4F50515259BFCF9FAFF D</td>
</tr>
<tr>
<td>5CF7535455565758595AB2D6D6D23D5 E</td>
</tr>
<tr>
<td>30313233343536373839B3DBDC9DA9F F</td>
</tr>
</tbody>
</table>

# Outbound (file to network) translation table:
# ISO8859-1 -> IBM-1047,swaplfnl

<table>
<thead>
<tr>
<th>0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>000102039C09867F978D8E0B0C0D0E0F #0</td>
</tr>
<tr>
<td>1011112139D0A08871819928F1C1D1E1F #1</td>
</tr>
<tr>
<td>405A7F7B5B6C507D4D5D5C4E6B604B61 #2</td>
</tr>
<tr>
<td>F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F #3</td>
</tr>
<tr>
<td>C7C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6 #4</td>
</tr>
<tr>
<td>D7D8E9E2E3E4E5E6E7E8E9EDE0B5F6D #5</td>
</tr>
<tr>
<td>9781828384858687888991293949596 #6</td>
</tr>
<tr>
<td>979899A2A3A4A5A6A7A8A9A0A1A0A10 #7</td>
</tr>
<tr>
<td>2021222324252617282292A2B2C09A1B #8</td>
</tr>
<tr>
<td>3031A33333435363738393A3B04143E #9</td>
</tr>
<tr>
<td>41AA4AB19FB26AB5BBB49A8AB0AAFBC A</td>
</tr>
<tr>
<td>908F4AFAEB0B6B39DDA9BB88B8B89AB B</td>
</tr>
<tr>
<td>64656266636768697172737757677C C</td>
</tr>
<tr>
<td>AC69EDEEEFECBF80FDFFBCBAAE59 D</td>
</tr>
<tr>
<td>44454246434749484541525358555657 E</td>
</tr>
<tr>
<td>8C49CDCEBCFC6EC170DDDEDBD8C88EDF F</td>
</tr>
</tbody>
</table>

EOF
Note

When ICU libraries are used for generating ASCII to EBCDIC translation tables, \texttt{,swaplfnl} must be added to the EBCDIC codepage name so that ASCII line feed characters (0A) are correctly translated to EBCDIC newline characters (15).

In order to create a custom translation table, first create a translation table with \texttt{ssh-translation-table} and then edit it with any text editor.

Environment Variables

\texttt{SSH\_CHARSET\_CONV}

The full pathname of the Tectia conversion DLL. Only required if \texttt{ssh-translation-table} or the conversion DLL are not in the installation directories. Here is an example of the pathname:

\texttt{SSH\_CHARSET\_CONV=/opt/tectia/lib/shlib/i18n\_iconv.so}
ssh-sft-stage

ssh-sft-stage -- stage and destage MVS data sets and HFS files

Synopsis


Description

ssh-sft-stage stages a file or data set, that is, converts it into transfer format, or destages a stage file, that is, converts it into a file or data set. File transfer advice strings and profiles can be used to control staging.

The Secure File Transfer Protocol (SFTP), and the way in which it is used in common Secure Shell implementations, requires that a file to be transferred is an octet sequence, that the file size is known, and that it is possible to seek to any position in the file. Staging transforms structured MVS data sets into such a format.

On z/OS Unix such files are known as Hierarchical File System (HFS) files.

There are three transfer formats:

- Line format is character-based and uses the newline character as a record delimiter.
- Record format includes record length fields in the stored or transferred data.
- Stream format stores records end-to-end and does not include any structural information.

Input files and output files may be HFS files or MVS data sets. Output MVS data sets must be pre-allocated with complete DCB parameters.

ssh-sft-stage performs coded character set (CCS) conversion when a CCS option is specified for both the stage file and the MVS data set. All the character sets known by the `iconv()` function are available. To see a list of the available character sets, run

```
> iconv -l
```

in the environment where you intend to run ssh-sft-stage.

The Tectia Server for IBM z/OS server program, ssdh2, uses temporary memory files as stage files. This mechanism can be verified with ssh-sft-stage by omitting the -s option and using -i only or both the -i and -o options.
Options

`ssh-sft-stage` has both a long and short form for the options. Both are shown below – use either of the forms.

`-i`, `--input-file=FILE`
Specifies the name of the input file to be staged.

`-s`, `--stage-file=FILE`
Specifies the stage file name.

`-o`, `--output-file=FILE`
Specifies the file name resulting from destaging the stage file.

`-I`, `--input-file-ccs=CCS`
Specifies the CCS of the input file.

`-S`, `--stage-file-ccs=CCS`
Specifies the CCS of the stage file.

`-O`, `--output-file-ccs=CCS`
Specifies the CCS of the output file.

`-f`, `--stage-format=FORMAT`
Specifies the format of the stage file: stream, line, or record.

`-b`, `--buffer-size=BUFFER_SIZE`
Specifies the size of file buffer (in bytes).

`-p`, `--profile`
Enables file transfer advice profiles.

`-d|-D`, `--debug=STRING`
Enables debugging.

`-v`, `--verbose`
Generates verbose output.

`-V`, `--version`
Displays program version and exits.

`-h`, `--help`
Displays a short summary of command-line options and exits.
Files

Data Set Names

Write MVS data set names with leading slashes, e.g. //HANDY.PROGRAM.

HFS file names must include at least one slash, but not two at the start of the name.

Use single quotes for data set names that should not be prefixed with the user's prefix:
"/SYS3.HANDY.PROGRAM"

Input and output files may also be members in PDS and PDSE libraries. Write the member name in parentheses. Use quoting to prevent the shell from trying to interpret the parentheses:
"/SYS3.CODE(HANDY)"

Output Data Sets

MVS output data sets must be pre-allocated. Any content will be overwritten.

When destaging, overrunning the data set record length (or block size for the U data set organization) will cause the run to fail. Overrun can be caused by excessive record lengths when the stage file format is record or missing or improperly placed newline characters when the stage file format is line.

For the line format, the stage file must be encoded in EBCDIC or must be converted to EBCDIC with CCS options. The newline character must be NL (0x15 in hexadecimal and 025 in octal).

When the stage file format is line, fixed length records are padded with blanks. When the stage file format is record, fixed length records are padded with binary zeros. For the stream format, the last record may be padded with binary zeros.

Environment Variables

You must set this environment variable:

_BPXK_AUTOCVT=ON

If this variable is not set correctly, ssh-sft-stage fails to start.

See the IBM z/OS C/C++ Run-Time Library Reference manual for these two variables, which influence how ssh-sft-stage performs character set conversions:

• _ICONV_UCS2
• _ICONV_UCS2_PREFIX
ssh-keygen-g3

ssh-keygen-g3 -- authentication key pair generator

Synopsis

ssh-keygen-g3 [ options ...]
[ key1 key2 ...]

Description

ssh-keygen-g3 is a tool that generates and manages authentication keys for Secure Shell. Each user wishing to use a Secure Shell client with public-key authentication can run this tool to create authentication keys. Additionally, the system administrator can use this to generate host keys for the Secure Shell server. This tool can also convert openSSH public or private keys to the Tectia key format, or, from Tectia key format to openSSH format. Tectia public keys use The Secure Shell (SSH) Public Key File Format (RFC 4716).

By default, if no path for the key files is specified, the key pair is generated under the user's home directory ($HOME/.ssh2 on Unix, "%APPDATA%\SSH\UserKeys" on Windows). If no file name is specified, the key pair is likewise stored under the user's home directory with such file names as id_rsa_2048_a and id_rsa_2048_a.pub.

When specifying file paths or other strings that contain spaces, enclose them in quotation marks ("").

Options

The following options are available:

-1 file
  Converts a key file from the SSH1 format to the SSH2 format. Note: "1" is number one (not letter L).

-7 file
  Extracts certificates from a PKCS #7 file.

-b bits
  Specifies the length of the generated key in bits. The allowed and default lengths for different key types are:
  • RSA/DSA: allowed 512 to 65536 bits, default 2048 bits
  • ECDSA: allowed 256, 384 and 521 bits, default 256 bits

-B num
  Specifies the number base for displaying key information (default: 10).
-c comment
   Specifies a comment string for the generated key.

-D file
   Derives the public key from the private key file.

-e file
   Edits the specified key. Makes ssh-keygen-g3 interactive. You can change the key’s passphrase or comment.

-F, --fingerprint file

   Dumps the fingerprint and type (RSA, DSA or ECDSA) of the given public key. By default, the fingerprint is given in the SSH Babble format, which makes the fingerprint look like a string of "real" words (making it easier to pronounce). The output format can be changed with the --fingerprint-type option.

   The following options can be also used to modify the behavior of this option: --fingerprint-type --hash, --hostkeys-directory, --known-hosts, --rfc4716.

-F, --fingerprint host_ID

   Dumps the location, fingerprint and type (RSA, DSA or ECDSA) of the locally stored host key(s) identified with the given host_ID. The host_ID is a host name or string "host#port".

   The following options can be used to modify the behavior of this option: --fingerprint-type, --hash, --hostkeys-directory, --known-hosts, --rfc4716.

-H, --hostkey

   Stores the generated key pair in the default host key directory (/opt/tectia/etc). Specify the -P option to store the private key with an empty passphrase.

-l file

   Loads and displays information on the key file.

-k file

   Converts a PKCS #12 file to an SSH2-format certificate and private key.

-m, --generate-moduli-file


-p passphrase

   Specifies the passphrase for the generated key.

-P

   Specifies that the generated key will be saved with an empty passphrase.

-q, --quiet

   Hides the progress indicator during key generation.
-r file
   Adds entropy from file to the random pool. If file contains ‘relatively random’ data (i.e. data unpredictable by a potential attacker), the randomness of the pool is increased. Good randomness is essential for the security of the generated keys.

-t dsa | rsa | ecdsa
   Selects the type of the key. Valid values are rsa (default), dsa and ecdsa.

-x file
   Converts a private key from the X.509 format to the SSH2 format.

--append [ -yes | no ]
   Appends the keys. Optional values are yes and no. The default is yes to append.

--copy-host-id host_ID destination
   Copies the host identity to the specified destination directory.

   The following options can be used to modify the behavior of this option: --append, --hostkeys-directory, --known-hosts, --overwrite.

   If --hostkey-file is given, the file is treated as a normal host identity file used by the Connection Broker, and its contents will be copied to the destination directory.

--delete-host-id host_ID
   Deletes the host key of the specified host ID. The host_ID is a host name or string ”host#port”.

   The following options can be used to modify the behavior of this option: --host-key-file, --hostkeys-directory, --known-hosts.

--fingerprint-type babble | babble-upper | pgp-2 | pgp-5 | hex | hex-upper
   Specifies the output format of the fingerprint. If this option is given, the -F option and the key file name must precede it. The default format is babble.

   See the section called “Examples” for examples of using this option.

--hash md5 | sha1
   Specifies the digest algorithm for fingerprint generation. Valid options are md5 and sha1.

--hostkey-file file
   When copying, uses the given file as the source host key, instead of autodetecting the location. When deleting, only deletes from the given location. If the specified file does not contain identities for the specified host, does nothing.

--hostkeys-directory directory
   Specifies the directory for known host keys to be used instead of the default location.
--import-public-key infile [outfile]
Attempts to import a public key from infile and store it to outfile in the format specified by --key-format parameter. If outfile is not given, it will be requested. The default output format is SSH2 native format.

--import-private-key infile [outfile]
Attempts to import a private key from infile and store it to outfile in the format specified by --key-format parameter. If outfile is not given, it will be requested. The default output format is SSH2 native private key format.

--import-ssh1-authorized-keys infile outfile
Imports an SSH1-style authorized_keys file infile and generates an SSH2-style authorization file outfile, and stores the keys from infile to generated files into the same directory with outfile.

--key-format format
Output key format: secsh2, pkcs1, pkcs8, pkcs12, openssh2, or openssh2-aes.

--key-hash hash
This option can be used for other than Tectia key formats. Specifies the hash algorithm to be used in passphrase-based private key derivation. The default value is sha1. Other supported algorithms are sha224, sha256, sha384, and sha512. Note that all key formats do not support all hash algorithms.

--known-hosts file
Uses the specified known hosts file. Enables fetching fingerprints for hosts defined in an OpenSSH-style known-hosts file. Using this option overrides the default locations of known_hosts files (/etc/ssh/ssh_known_hosts and $HOME/.ssh/known_hosts). Giving an empty string will disable known-hosts usage altogether.

--moduli-file-name file
Writes the moduli generated for Diffie-Hellman group exchange to file. (The default file name for option -m is secsh_dh_gex_moduli.)

--overwrite [ -yes ] no
Overwrite files with the same file names. The default is to overwrite.

--rfc4716
Displays the fingerprint in the format specified in RFC4716. The digest algorithm (hash) is md5, and the output format is the 16-bytes output in lowercase HEX separated with colons (:).

-v
Displays version string and exits.

-h, --help, -?
Displays a short summary of command-line options and exits.
Environment Variables

In order to run `ssh-keygen-g3` the following environment variables must be set:

- `_BPXK_AUTOCVT` = ON
  
  If this variable is not set correctly `ssh-keygen-g3` fails to start.

- `SSH_ZCPUMATH_FACILITY`

  Controls the z13 Vector Facility usage. The default value is 1. If this variable is set to 1, `ssh-keygen-g3` will probe the availability of the z13 Vector Facility. If the facility is available, `ssh-keygen-g3` will use it. If this variable is set to 0 or the facility is not available, `ssh-keygen-g3` will not use the z13 vector facility.

Examples

Create a 3072-bit RSA key pair and store the key pair in the default user key directory with file names `newkey` and `newkey.pub`:

```bash
$ ssh-keygen-g3 -b 3072 newkey
```

Convert an SSH1 key `oldkey` to SSH2 format:

```bash
$ ssh-keygen-g3 -1 oldkey
```

Display the fingerprint of a server host public key in SSH babble (default) format:

```bash
$ ssh-keygen-g3 -F hostkey.pub
Fingerprint for key:
xeneh-fyvam-sotaf-gutuv-rahih-kipod-poten-byfam-hufeh-tydym-syxex
```

Display the fingerprint of a server host public key in hex format:

```bash
$ ssh-keygen-g3 -F hostkey.pub --fingerprint-type=hex
Fingerprint for key:
25533b8c7734f6eb1556ea2ab4900d854d5d088c
```

Convert a private key into openSSH2-AES format:

```bash
$ ssh-keygen-g3 -p <password> --key-format openssh2-aes \ --import-private-key <source_key_file> <destination_key_file>
```

Note: if the private key file that is being converted is encrypted with a passphrase, the passphrase must be provided with the ‘-p’ option.

Convert a Tectia public key `tectiakey.pub` to an OpenSSH public key `opensshkey.pub`:

```bash
$ ssh-keygen-g3 --key-format openssh2 --import-public-key \ tectiakey.pub opensshkey.pub
```

Generate moduli file `dhgex-moduli` for Diffie-Hellman group exchange:
$ ssh-keygen-g3 -m --moduli-file-name dhgex-moduli
ssh-keydist-g3

ssh-keydist-g3 -- Key distribution tool

Synopsis

ssh-keydist-g3 [options...] host [ [options...][host] ...]

Description

The ssh-keydist-g3 key distribution tool can be used for storing multiple remote host keys to a common key store and setting up public-key authentication to multiple hosts.

The tool uses sub-script ssh-keyfetch for fetching remote host keys.

The tool calls ssh-keygen-g3 when creating new key pairs.

Options

ssh-keydist-g3 accepts the following options:

- A, --accepted-host-key-log FILE
  Specifies a log file listing the accepted new host keys. The default is ssh_host_keys.log in the user home directory.

- b, --key-bits NUMBER
  Specifies the length of the generated key in bits (default 2048).

- d, --allow-keygen-overwrite
  Allows ssh-keygen-g3 to overwrite an existing key pair.

- D, --debug LEVEL
  Sets the debug level, where LEVEL is number from 1 to 99.

- f, --pubkey-file FILE
  Disables key pair generation, and distributes the given key file instead.

- F, --accepted-host-key-filename-format plain|hashed
  The accepted host keys are stored in the specified filename format. The default is hashed. See Section Section 4.2.1 for more information.

- g, --accept-hostkeys-globally
  The accepted host keys are copied to the system-wide store for trusted host keys (/opt/tectia/etc/hostkeys). This causes all users to trust the host key. Giving this option requires administrator privileges.
-H, --hostlist-file FILE
   Specify a host list file that contains hostnames or username/hostname pairs.

   The format of the host list file is as follows:
   
   userid1/host1.example.com,passwordfile1
   userid2/host2.example.com,passwordfile2
   userid3/host3.example.com,passwordfile3

   If the user name is omitted from the entry, the user name given with the -u option is used for the connection. If -u has not been given, the local user name is used.

   If the password file is omitted from the entry, the password file given with the -p option is used for the connection. If -p has not been given, the password is prompted interactively from the user.

-i, --accept-host-keys-also-by-ip
   Stores the accepted host keys also by their IP address. This option must be specified if the host will be accessed with Transparent FTP tunneling.

-I, --dont-accept-host-keys-also-by-ip
   Does not store the accepted host keys also by their IP address (default).

-k, --continue-after-error
   Do not exit if an operation for one host fails but continue with other hosts.

-l, --accept-hostkeys-locally
   The accepted host keys are copied to the user specific store for accepted keys. This is the default.

-n, --do-not-execute
   Prints the commands but does not execute them.

-N, --accept-host-keys
   Accepts new host keys. Does not generate or distribute user keys.

-O, --openssh-unix
   The remote host is running Unix and its Secure Shell server is OpenSSH. The public key is appended to the user's $HOME/.ssh/authorized_keys file.

-p, --password-file FILE
   Specify a file or a data set containing the password for authenticating to remote server(s) during public key setup. Use with care!

-P, --empty-passphrase
   Generate the key pair with an empty passphrase.

-S, --ssh2-unix
   The remote host is running Unix and its Secure Shell server is Tectia. The public key is uploaded to the user's $HOME/.ssh2 directory and the $HOME/.ssh2/authorization file is updated.
-t, --key-type dsa|rsa|ecdsa
   Selects the algorithm used in key generation. rsa (default), dsa and ecdsa are supported.

-u, --remote-user USER
   Specify remote user name. The default is the local user name.

-U, --user-key-log FILE
   Specifies a log file listing the generated and distributed user keys. The default is ssh_user_keys.log in the user home directory.

-v, --verbose
   Enables verbose mode. Information on the progress of the program is displayed in standard output.

-W, --ssh2-windows
   The remote host is running Windows and its Secure Shell server is Tectia. The public key is uploaded to the user's %USERPROFILE%\.ssh2 directory and the %USERPROFILE%\.ssh2\authorization file is updated.

-Z, --ssh2-zos
   The remote host is running z/OS and its Secure Shell server is Tectia. The public key is uploaded to the user's USS $HOME/.ssh2 directory and the $HOME/.ssh2/authorization file is updated.

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 by verifying the key fingerprints after receiving them or you risk being subject to a man-in-the-middle attack.

To validate the host key, obtain the host key fingerprint from a trusted source (for example by calling the server administrator) and verify it against the output from command:

```
ssh-keygen-g3 --fingerprint <hostname>
```

Examples

Example 1: Connect to multiple hosts, fetch their host keys in hashed (default) format, and save them under the user's $HOME/.ssh2/hostkeys directory. Save the host key hash values with both the specified hostname and the IP address of the host. Store a log of the accepted new host keys under /tmp.

```
$ ssh-keydist-g3 -N -i -A /tmp/newhosts.log host1 host2 host3
```

Example 2: Connect to multiple hosts defined in the hostlist.txt file, fetch their host keys in plain format, and save them under the system-wide /opt/tectia/etc/hostkeys directory. Running the command requires administrator privileges.

```
# ssh-keydist-g3 -N -F plain -g -H /home/userid/hostlist.txt
```
The keys are stored with the names specified in the host list file. For example, the following list would specify storing the keys with FQDN and also connecting to port 222 on host1.example.com:

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>host1.example.com</td>
<td>-</td>
</tr>
<tr>
<td>host2.example.com</td>
<td>-</td>
</tr>
<tr>
<td>host3.example.com</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example 3:** Create a 2048-bit DSA key with an empty passphrase, and upload it to a Unix server running OpenSSH, including the necessary conversions. Public-key upload uses password-from-file for authentication.

```bash
$ ssh-keydist-g3 -t dsa -b 2048 -p -d -p /home/userid/passwd_file \
  -u user1 -O open_server.example.com
```

**Example 4:** Create a 2048-bit RSA key with an empty passphrase, and upload it to multiple servers, including the necessary conversions. Public-key upload uses password-from-file for authentication. `passwd_file1` is used for the Unix, Windows, and z/OS hosts running Tectia and `passwd_file2` is used for the host running OpenSSH.

```bash
$ ssh-keydist-g3 -t -P -d \
  -p /home/userid/passwd_file1 \
  -S -u user1 tectia_unix.example.com \
  -W -u user2 tectia_win.example.com \
  -Z -u user3 tectia_zos.example.com \
  -p /home/userid/passwd_file2 \
  -O -u user1 open_server.example.com \
```

**Example 5:** Distribute an existing RSA public key to several hosts using host lists. Store the log of distributed keys under `/tmp`.

The host lists need to be grouped so that all Tectia Unix, Tectia Windows, Tectia z/OS, and OpenSSH hosts are in different host files, for example `tectiaunix_hostlist.txt`, `tectiazos_hostlist.txt`, `openssh_hostlist.txt`, each host list defined in the following way:

<table>
<thead>
<tr>
<th>Host Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid1/host1.example.com</td>
</tr>
<tr>
<td>userid2/host2.example.com</td>
</tr>
<tr>
<td>userid3/host3.example.com</td>
</tr>
</tbody>
</table>

The command is as follows:

```bash
$ ssh-keydist-g3 -f /home/userid/.ssh2/id_rsa_2048_a.pub \
  -p /home/userid/common_passwd_file -F plain -U /tmp/userkeys.log \
  -S -H tectiaunix_hostlist.txt \
  -Z -H tectiazos_hostlist.txt \
  -O -H openssh_hostlist.txt
```
ssh-keyfetch

ssh-keyfetch -- Host key tool for the Secure Shell client

Synopsis

ssh-keyfetch [options...] [host]

Description

ssh-keyfetch is a tool that downloads server host keys and optionally sets them as known host keys for the Secure Shell client. It is typically used by the system administrator during the initial setup phase.

By default the host key is fetched from the server and saved in file key_host_port.suffix in the current directory.

Options

The following options are available:

-a, --set-trusted

Instead of writing the public key to a file, add the public key as a known host key to the user-specific directory: $HOME/.ssh2/hostkeys. This option cannot be combined with -C or -K.

Caution

When ssh-keyfetch is run with the -a option, it accepts the received host keys automatically without prompting the user. You should verify the validity of keys by verifying the key fingerprints after receiving them or you risk being subject to a man-in-the-middle attack.

To validate the host key, obtain the host key fingerprint from a trusted source (for example by calling the server administrator) and verify it against the output from command:

```
ssh-keygen-g3 --fingerprint <hostname>
```

-A, --fetch-any

Probe for and fetch either server public key or certificate.

-C, --fetch-certificate

Probe for and fetch the server certificate only.

-d, --debug debug-level

Enable debugging.
-D, --debug-default
  Enable debugging with default level.

-f, --filename-format nameformat
  Filename format for known host keys. Accepted values are plain and hashed. The default is plain.

-F, --fingerprint-type [-babble babble-upper | pgp-2 | pgp-5 | hex | hex-upper]
  Public key fingerprint type for fingerprints displayed in messages and log. Most popular types are babble (the SSH babble format) and hex. The default is babble. See also the option --rfc4716.

-H, --hash [-md5 | sha1]
  Specifies the digest algorithm for fingerprint generation. Valid options are md5 and sha1.

-K, --kex-key-formats typelist
  Explicitly specify the host-key types accepted in protocol key exchange. For experts only. See RFC 4253 for details.

-l, --log
  Report successfully received keys in log format. The log format consists of one line per key, six fields per line. The fields are:
  • accept|save
  • replace|append
  • hostname
  • ip-port
  • user-id
  • key-file-path
  • fingerprint

-o, --output-file output-file
  Write result to output-file. A minus sign ("-") denotes standard output.

-O, --output-directory output-dir
  Write result to output-dir. The default is the current directory.

-p, --port port
  Server port (default: 22).

-P, --fetch-public-key
  Probe for and fetch the server public key only. This is the default behaviour.
-q, --quiet
  Quiet mode, report only errors.

-R, --rfc4716
  Displays the public key fingerprints in the format specified in RFC 4716. The digest algorithm (hash) is md5, and the output format is the 16-bytes output in lowercase HEX separated with colons (:).

-S, --proxy-url socks-url
  Specifies the SOCKS server to use.

-t, --timeout timeout
  Connection timeout in seconds (default: 10 seconds).

--append [ --yes | no ]
  Instead of appending a new host key, overwrite the existing known host keys for this host. Optional values are yes and no. The default is to append.

-V, --version
  Displays version string and exits.

Environment Variables

In order to run ssh-keyfetch the following environment variables must be set:

_BPXK_AUTOCVT=ON
  If this variable is not set correctly ssh-keyfetch fails to start.

SSH_SOCKS_SERVER
  The address of the SOCKS server used by ssh-keyfetch.

Examples

Connect to the server through a SOCKS proxy:

```bash
$ ssh-keyfetch -S socks://fw.example.com:1080/10.0.0.0/8 server.outside.example
Public key from server.outside.example:22 saved.
  File: server.outside.example.pub
```

Accept the server key as a known key for Tectia Client and report in the more rigid log format:

```bash
$ ssh-keyfetch -a -l newhost
Accepted newhost 22 testuser /home/testuser/.ssh2/hostkeys/key_22_newhost.pub
  xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuv-muled-syxyx
```

Accept the server key as a known key for Tectia client tools for z/OS and store the key to global configuration hostkeys directory:
Accept the server key as a known key for Tectia Client and use an uninformative hash as the filename for the stored known key:

```
$ ssh-keyfetch -a --output-directory /etc/ssh2/hostkeys
Accepted newhost 22 testuser /etc/ssh2/hostkeys/key_22_anotherhost.pub
bydop-mulym-zegar-nybuv-muled-syxyx-xigad-hozuf-kykek-vogid-dumid
```

```
$ ssh-keyfetch -f hashed -a newhost
Public key from newhost:22 accepted as trusted hostkey.
File: /home/testuser/.ssh2/hostkeys/keys_420b23ca959ab165e52e117a90baa89d92ffc535
Fingerprint: xigad-hozuf-kykek-vogid-dumid-bydop-mulym-zegar-nybuv-muled-syxyx
```

Accept RSA and ECDSA server keys as a known key for Tectia Client:

```
$ for t in ecdsa rsa; do
  ssh-keyfetch --set-trusted -k $t newhost
done
Public key from newhost accepted as trusted hostkey.
File: /home/testuser/.ssh2/hostkeys/key_22_newhost.pub
Public key from newhost:22 accepted as trusted hostkey.
File: /home/testuser/.ssh2/hostkeys/key_22_newhost.pub
Fingerprint: xuzib-sehat-pemys-zulor-foran-tizur-repyh-boryd-nogeb-refip-raxax
Public key from newhost:22 accepted as trusted hostkey.
File: /home/testuser/.ssh2/hostkeys/key_22_newhost.pub
Fingerprint: xifon-sorer-pysys-vumab-mosuz-pefor-pezab-givaz-feguc-nyven-lexux
```

Fetch the X.509 certificate of the server running in port 222 and display the content with `ssh-certview`:

```
$ ssh-keyfetch -C -p 222 -o - newhost | ssh-certview -
Certificate =
  SubjectName = <C=FI, O=SSH, OU=DEV, CN=newhost.ssh.com>
  IssuerName = <C=FI, O=SSH, CN=Sickle CA>
  SerialNumber= 24593438
  Validity =
    NotBefore = 2007 Sep 13th, 15:10:00 GMT
    NotAfter = 2008 Sep 12th, 15:10:00 GMT
PublicKeyInfo =
  PublicKey =
    Algorithm = RSA
    Modulus n (1024 bits):
      ...
Fingerprints =
  SHA-1 =
```
This explicit key exchange type list is equivalent to specifying option -A:

```bash
$ ssh-keyfetch -K
ssh-rsa-sha512@ssh.com,ssh-rsa-sha384@ssh.com,ssh-rsa-sha256@ssh.com,ssh-rsa-sha224@ssh.com,ssh-rsa,ssh-dss-sha512@ssh.com ...
Public key from newhost:22 saved.
File: key_newhost_22.pub
Fingerprint:
xigad-hozuf-kyek-vogid-bydop-mulym-zegar-nybuf-fully-syxyx
```
ssh-cmpclient-g3

ssh-cmpclient-g3 -- CMP enrollment client

Synopsis

ssh-cmpclient-g3 command [options] access [name]

Where command is one of the following:

INITIALIZE psk|racerts keypair template
ENROLL certs|racerts keypair template
UPDATE certs [keypair]
POLL psk|certs|racerts id

RECOVER psk|certs|racerts template
REVOKE psk|certs|racerts template

TUNNEL racerts template

Most commands can accept the following options:

-B            Perform key backup for subject keys.
-o prefix     Save result into files with given prefix.
-O filename   Save the result into the specified file.
              If there is more than one result file,
              the remaining results are rejected.
-C file       CA certificate from this file.
-S url        Use this SOCKS server to access the CA.
-H url        Use this HTTP proxy to access the CA.
-E            PoP by encryption (CA certificate needed).
-v num        Protocol version 1|2 of the CA platform. Default is 2.
-y            Non-interactive mode. All questions answered with 'y'.
-N file       Specifies a file to stir to the random pool.
-d level      Set debug level.
-Z provspec   Specifies external key provider for the private key.
              The format of provspec is "providername:initstring".

The following identifiers are used to specify options:

psk               -p refnum:head (reference number and pre-shared key)
                  -p file (containing refnum:head)
                  -i number (iteration count, default 1024)

certs            -c file (certificate file) -k url (private-key URL)
racerts          -R file (RA certificate file) -k url (RA private-key URL)
keypair          -P url (private-key URL)
id                -I number (polling ID)
template         -T file (certificate template)
                  -s subject-ldap[;type=value]
                  -u key-usage-name[;key-usage-name]
                  -U extended-key-usage-name[;extended-key-usage-name]
access URL where the CA listens for requests.
name LDAP name for the issuing CA (if -C is not given).

Key URLs are either valid external key paths or in the format:
"generate://savetype:passphrase@keytype:size/save-file-prefix"
"file://passphrase/relative-key-file-path"
"file:relative-key-file-path"
"any-key-file-path"

The key generation "savetype" can be:
- ssh2, secsh2, secsh (Secure Shell 2 key type)
- ssh1, secsh1 (legacy Secure Shell 1 key type)
- pkcs1 (PKCS #1 format)
- pkcs8s (passphrase-protected PKCS #8, "shrouded PKCS #8")
- pkcs8 (plain-text PKCS #8)
- x509 (Tectia-proprietary X.509 library key type)

-h Prints usage message.
-F Prints key usage extension and keytype instructions.
-e Prints command-line examples.

Description

The ssh-cmpclient-g3 command-line tool is a certificate enrollment client that uses the CMP protocol. It can generate an RSA or DSA public-key pair and get certificates for their public components. CMP is specified by the IETF PKIX Working Group for certificate life-cycle management, and is supported by some CA platforms, such as RSA Keon.

Commands

The ssh-cmpclient-g3 command-line command keywords are listed below. Shorthands longer than three letters can be used to identify the command. The commands are case-insensitive. The user must specify the CA address URL for each command. Here the term "user" refers to a user, program, or hardware device.

INITIALIZE
Requests the user's initial certificate. The request is authenticated using the reference number and the corresponding key (PSK) received from the CA or RA using some out-of-band mechanism.

The user must specify the PSK, the asymmetric key pair, and a subject name.

ENROLL
Requests a new certificate when the user already has a valid certificate for the key. This request is similar to initialize except that it is authenticated using public-key methods.

POLL
Polls for a certificate when a request was not immediately accepted.
UPDATE
Requests an update of an existing certificate (replacement). The issued certificate will be similar to the existing certificate (names, flags, and other extensions). The user can change the key, and the validity times are updated by the CA. This request is authenticated by a valid existing key pair and a certificate.

RECOVER
Requests recovery of a backed-up key. This request is authenticated either by PSK-based or certificate-based authentication. The template describes the certificate whose private key has already been backed up and should be recovered. Users can only recover keys they have backed up themselves.

REVOKE
Requests revocation for a key specified in the template. Authentication of the request is made using a PSK or a certificate belonging to the same user as the subject of revocation.

TUNNEL
Operates in RA tunnel mode. Reads requests and optionally modifies the subject name, alternative names, and extensions based on the command line. Approves the request and sends it to the CA.

Options
The **ssh-cmpclient-g3** command-line options are listed below. Note that when a file name is specified, an existing file with the same name will be overwritten. When specifying subject names or other strings that contain spaces, enclose them in quotation marks ("").

-B
Requests private key backup to be performed for the initialize, enroll, and update commands.

-o prefix
Saves resulting certificates and CRLs into files with the given prefix. The prefix is first appended by a number, followed by the file extension .crt or .crl, depending on the type of object.

-O filename
Saves the result into the specified absolute filename. If there is more than one result file, the remaining results are rejected.

-C file
Specifies the file path that contains the CA certificate. If key backup is done, the file name must be given, but in most cases the LDAP name of the CA can be given instead.

-S url
Specifies the SOCKS URL if the CA is located behind a SOCKS-enabled firewall. The format of the URL is: socks://[username@]server[:port][/network/bits[,network/bits]]

-H url
Uses the given HTTP proxy server to access the CA. The format of the URL is: http://server[:port]/
-E
Performs encryption proof of possession if the CA supports it. In this method of PoP, the request is not signed, but instead the PoP is established based on the ability to decrypt the certificates received from the CA. The CA encrypts the certificates with the user's public key before sending them to the user.

-v num
Selects the CMP protocol version. This is either value 1, for an RFC 2510-based protocol, or 2 (the default) for CMPv2.

-N file
Specifies a file to be used as an entropy source during key generation.

-d level
Sets the debug level string to level.

-Z provspec
Specifies the external key provider for the private key. Give provspec in the format "providername:init-string".

The usage line uses the following meta commands:

psk
The reference number and the corresponding key value given by the CA or RA.

-p refnum:key|file
refnum and key are character strings shared among the CA and the user. refnum identifies the secret key used to authenticate the message. The refnum string must not contain colon characters.

Alternatively, a filename containing the reference number and the key can be given as the argument.

-i number
number indicates the key hashing iteration count.

certs
The user's existing key and certificate for authentication.

-k url
URL specifying the private key location. This is an external key URL whose format is specified in the section called “Synopsis”.

-c file
Path to the file that contains the certificate issued to the public key given in the -k option argument.

racerts
In RA mode, the RA key and certificate for authentication.
-k url
    URL specifying the private key location. This is an external key URL whose format is specified in the section called “Synopsis”.

-R file
    Path to the file that contains the RA certificate issued to the public key given in the -k option argument.

keypair
    The subject key pair to be certified.

-P url
    URL specifying the private key location. This is an external key URL whose format is specified in the section called “Synopsis”.

id
    Polling ID used if the PKI action is left pending.

-I number
    Polling transaction ID number given by the RA or CA if the action is left pending.

template
    The subject name and flags to be certified.

-T file
    The file containing the certificate used as the template for the operation. Values used to identify the subject are read from this, but the user can overwrite the key, key-usage flags, or subject names.

-s subject-ldap[type=value]*
    A subject name in reverse LDAP format, that is, the most general component first, and alternative subject names. The name subject-ldap will be copied into the request verbatim.

    A typical choice would be a DN in the format "C=US,O=SSH,CN=Some Body", but in principle this can be anything that is usable for the resulting certificate.

    The possible type values are ip, email, dn, dns, uri, and rid.

-u key-usage-name[key-usage-name]*
    Requested key usage purpose code. The following codes are recognized: digitalSignature, non-Repudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly, decipherOnly, and help. The special keyword help lists the supported key usages which are defined in RFC 3280.

-U extended-key-usage-name[;extended-key-usage-name]*
    Requested extended key usage code. The following codes, in addition to user-specified dotted OID values are recognized: serverAuth, clientAuth, codeSigning, emailProtection, timeStamping, ikeIntermediate, and smartCardLogon.
access
Specifies the CA address in URL format. Possible access methods are HTTP (http://host:port/path), or plain TCP (tcp://host:port/path). If the host address is an IPv6 address, it must be enclosed in square brackets (http://[IPv6-address]:port/).

name
Optionally specifies the destination CA name for the operation, in case a CA certificate was not given using the option -C.

Examples
Initial Certificate Enrollment

This example provides commands for enrolling an initial certificate for digital signature use. It generates a private key into a PKCS #8 plaintext file named initial.prv, and stores the enrolled certificate into file initial-0.crt. The user is authenticated to the CA with the key identifier (refnum) 62154 and the key ssh. The subject name and alternative IP address are given, as well as key-usage flags. The CA address is pki.ssh.com, the port 8080, and the CA name to access Test CA 1.

```
$ ssh-cmpclient-g3 INITIALIZE \
   -P generate://pkcs8@rsa:1024/initial -o initial \
   -p 62154:ssh \
   -s 'C=FI,O=SSH,CN=Example/initial;IP=1.2.3.4' \
   -u digitalsignature \
   http://pki.ssh.com:8080/pkix/ \n   'C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities'
```

As a response the command presents the issued certificate to the user, and the user accepts it by typing yes at the prompt.

Certificate =
   SubjectName = <C=FI, O=SSH, CN=Example/initial>
   IssuerName = <C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities>
   SerialNumber= 8017690
   SignatureAlgorithm = rsa-pkcs1-sha1
   Validity = ...
   PublicKeyInfo = ...
   Extensions =
      Viewing specific name types = IP = 1.2.3.4
   KeyUsage = DigitalSignature
   CRLDistributionPoints = ...
   AuthorityKeyID =
   SubjectKeyID =
   Fingerprints =
Key update

Before the certificate expires, a new certificate with updated validity period should be enrolled. `ssh-cmpclient-g3` supports key update, where a new private key is generated and the key update request is authenticated with the old (still valid) certificate. The old certificate is also used as a template for issuing the new certificate, so the identity of the user will not be changed during the key update. With the following command you can update the key pair, which was enrolled in the previous example. Presenting the resulting certificate has been left out.

```bash
$ ssh-cmpclient-g3 UPDATE \
  -k initial.prv -c initial-0.crt -P \n  generate://pkcs8@rsa:1024/updatedcert -o updatedcert \n  http://pki.ssh.com:8080/pkix/ \n  "C=FI, O=SSH Communications Security Corp, CN=SSH Test CA 1 No Liabilities"
```

The new key pair can be found in the files with the `updatedcert` prefix. The policy of the issuing CA needs to also allow automatic key updates if `ssh-cmpclient-g3` is used in the `UPDATE` mode.
ssh-scepclient-g3

ssh-scepclient-g3 -- SCEP enrollment client

Synopsis

ssh-scepclient-g3 command [options] access [name]

Where command is one of the following:

- GET-CA
- GET-CHAIN
- ENROLL psk keypair template

Most commands can accept the following options:

- `-o` prefix Save result into files with prefix.
- `-S` url Use this socks server to access CA.
- `-H` url Use this HTTP proxy to access CA.

The following identifiers are used to specify options:

- psk `--p key` (used as revocationPassword or challengePassword)
- keypair `--P url` (private-key URL)
- ca `--C file` (CA certificate file)
- `--E file` (RA encryption certificate file)
- `--V file` (RA validation certificate file)
- template `--T file` (certificate template)
- `--s subject-ldap[;type=value]`
- `--u key-usage-name[;key-usage-name]`
- `--U extended-key-usage-name[;extended-key-usage-name]`

access URL where the CA listens for requests.

GET-CA and GET-CHAIN take name argument, that is something interpreted by the CA to specify a CA entity managed by the responder.

Key URLs are either valid external key paths or in the format:

- "generate://savetype:password@keytype:size/save-file-prefix"
- "file://savetype:password@/file-prefix"
- "file://passphrase/file-prefix"
- "file://file-prefix"
- "key-filename"

The "savetype" for the SCEP protocol has to be "rsa".

The key generation "savetype" can be:

- ssh2 (Secure Shell 2 key type)
- ssh1 (Legacy Secure Shell 1 key type)
- ssh (Tectia proprietary crypto library format, passphrase-protected)
- pkcs1 (PKCS#1 format)
- pkcs8s (passphrase-protected PKCS#8, "shrouded PKCS#8")
- pkcs8 (plain-text PKCS#8)
- x509 (Tectia proprietary X.509 library key type)
Description

The `ssh-scepclient-g3` command-line tool is a certificate enrollment client that uses the SCEP protocol. It can generate an RSA public-key pair and get certificates for its public components. The SCEP protocol was developed by Cisco and Verisign to be used on Cisco routers. Nowadays most CA platforms support this protocol for client certificate enrollment.

Commands

The `ssh-scepclient-g3` command-line command keywords are listed below. Shorthands longer than three letters can be used to identify the command. The commands are case-insensitive. The user must specify the CA address URL for each command. Here the term "user" refers to a user, program, or hardware device.

GET-CA

Requests CA or RA certificate download from the CA, and display the certificate fingerprint for CA validation. Fingerprints should be received from the CA using some out-of-band mechanism.

GET-CHAIN

Requests certificate chain from the CA/RA to the top-level CA.

ENROLL

Requests a new certificate from the CA. The CA will authorize the request using some out-of-band mechanism, or it can contain a password received from the CA.

Options

-o prefix

Saves output certificates into files with the given prefix. The prefix is first appended by a number, followed by the file extension .ca for CA certificates or .crt for user certificates.

-S url

Specifies the SOCKS URL if the CA is located behind a SOCKS-enabled firewall. The format of the URL is: socks://[username@]server[:port][/network/bits[,network/bits]]

-H url

Uses the given HTTP proxy server to access the CA. The format of the URL is: http://server[:port]/.

The usage line uses the following meta commands:

psk

The pre-shared key given by the CA or RA, or a revocation password invented by the client and provided to the CA when the user wishes to revoke the certificate issued. The type and need for this depends on the PKI platform used by the CA.
-p key
   An authentication password or a revocation password transferred (in encrypted format) to the CA
   for certification request or revocation request authorization purposes.

keypair
   The subject key pair to be certified.

-P url
   URL specifying the private key location. This is an external key URL whose format is specified in
   the section called “Synopsis”.

c a
   The CA/RA certificates.

-C file
   When performing enrollment, reads the CA certificate from the given file path.

-E file
   Optionally specifies the RA encryption certificate.

-V file
   Optionally specifies the RA signing certificate.

template
   The subject name and flags to be certified.

-T file
   The file containing the certificate used as the template for the operation. Values used to identify the
   subject are read from this, but the user may overwrite the key, key-usage flags, or subject names.

-s subject-ldap[;type=value]*
   A subject name in reverse LDAP format, that is, the most general component first, and alternative
   subject names. The name subject-ldap will be copied into the request verbatim.

   A typical choice would be a DN in the format "C=US,O=SSH,CN=Some Body", but in principle this
   can be anything that is usable for the resulting certificate.

   The possible type values are ip, email, dn, dns, uri, and rid.

-u key-usage-name[;key-usage-name]*
   Requested key usage purpose code. The following codes are recognized: digitalSignature, non-
   Repudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, 
   encipherOnly, decipherOnly, and help. The special keyword help lists the supported key usages
   which are defined in RFC 3280.
Requested extended key usage code. The following codes, in addition to user-specified dotted OID values are recognized: serverAuth, clientAuth, codeSigning, emailProtection, timeStamping, ikeIntermediate, and smartCardLogon.

access

Specifies the address of the CA in URL format. If the host address is an IPv6 address, it must be enclosed in brackets (http://[IPv6-address]:port/).

name

Specifies the destination CA name.

Examples

In the following example we first receive the CA certificate. The CA address is pki.ssh.com, the port is 8080, and the CA name is test-ca1.ssh.com.

```
$ ssh-scepclient-g3 GET-CA \
   -o ca http://pki.ssh.com:8080/scep/ \
   test-ca1.ssh.com

Received CA/RA certificate ca-0.ca:

```

Next, we enroll an RSA certificate. The user is authenticated to the CA with the key ssh. The subject name and alternative IP address are given, as well as key-usage flags.

```
$ ssh-scepclient-g3 ENROLL \
   -C ca-0.ca -p ssh \
   -o subject -P generate://pkcs8:ssh@rsa:1024/subject \
   -s 'C=FI,O=SSH,CN=SCEP Example;IP=1.2.3.4' \
   -u digitalsignature \
   http://pki.ssh.com:8080/scep/

Received user certificate subject-0.crt:

```

$ ls subject*
subject-0.crt  subject.prv
ssh-certview-g3

ssh-certview-g3 -- certificate viewer

Synopsis

ssh-certview-g3
[options...] file
[options...] file ...

Description

The ssh-certview-g3 program is a simple command-line application, capable of decoding and showing X.509 certificates, CRLs, and certification requests. The command output is written to the standard output.

Options

The following options are available:

- `h`
  Displays a short help.

- `verbose`
  Gives more diagnostic output.

- `quiet`
  Gives no diagnostic output.

- `auto`
  The next input file type is auto-detected (default).

- `cert`
  The next input file is a certificate.

- `certpair`
  The next input file is a cross-certificate pair.

- `crmf`
  The next input file is a CRMF certification request.

- `req`
  The next input file is a PKCS #10 certification request.

- `crl`
  The next input file is a CRL.
-prv
   The next input file is a private key.

-pkcs12
   The next input file is a PKCS#12 package.

-ssh2
   The next input file is an SSH2 public key.

-spkac
   The next input file is a Netscape-generated SPKAC request.

-noverify
   Does not check the validity of the signature on the input certificate.

-autoenc
   Determines PEM/DER automatically (default).

-pem
   Assumes that the input file is in PEM (ASCII base-64) format. This option allows both actual PEM (with
   headers and footers), and plain base-64 (without headers and footers). An example of PEM header and
   footer is shown below:

   -----BEGIN CERTIFICATE-----
   encoded data
   -----END CERTIFICATE-----

-der
   Assumes that the input file is in DER format.

-hexl
   Assumes that the input file is in Hexl format. (Hexl is a common Unix tool for outputting binary files in
   a certain hexadecimal representation.)

-skip number
   Skips number bytes from the beginning of input before trying to decode. This is useful if the file contains
   some garbage before the actual contents.

-ldap
   Prints names in LDAP order.

-utf8
   Prints names in UTF-8.

-latin1
   Prints names in ISO-8859-1.
Outputs big numbers in base-10 (default).

Outputs big numbers in base-16.

Outputs big numbers in base-64.

Sets output width (number characters).

Example

For example, using a certificate downloaded from pki.ssh.com, when the following command is given:

```
$ ssh-certview-g3 -width 70 ca-certificate.cer
```

The following output is produced:

```
Certificate =
    SubjectName = <C=FI, O=SSH Communications Security Corp, CN=Secure Shell Test CA>
    IssuerName = <C=FI, O=SSH Communications Security Corp, CN=Secure Shell Test CA>
    SerialNumber= 34679408
    SignatureAlgorithm = rsa-pkcs1-sha1
    Certificate seems to be self-signed.
        * Signature verification success.
    Validity =
        NotBefore = 2003 Dec  3rd, 08:04:27 GMT
        NotAfter  = 2005 Dec  2nd, 08:04:27 GMT
    PublicKeyInfo =
        PublicKey =
            Algorithm name (SSH) : if-modn(sign{rsa-pkcs1-md5})
            Modulus n  (1024 bits) :
                9635680922805930263476549641957998756341022541202937865240553
                9374740946079473767424224071470837728840839320521621518323377
                3593102350415987252300817926769968881159896955490274368606664
                0759644131690750532665262618696466060377799358036735475902257
                6086098562919363963470926690162744258451983124575595926849551
                903
            Exponent e ( 17 bits) :
                65537
    Extensions =
        Available = authority key identifier, subject key identifier, key usage(critical), basic constraints(critical), authority information access
        KeyUsage = DigitalSignature KeyEncipherment KeyCertSign CRLSign [CRITICAL]
```
BasicConstraints =
    PathLength = 0
    cA = TRUE

[CRITICAL]
AuthorityKeyID =
    KeyID =
SubjectKeyID =
    KeyId =
AuthorityInfoAccess =
    AccessMethod = 1.3.6.1.5.5.7.48.1
    AccessLocation =
        Following names detected =
            URI (uniform resource indicator)
        Viewing specific name types =
            URI = http://pki.ssh.com:8090/ocsp-1/
Fingerprints =
    SHA-1 =
ssh-ekview-g3

ssh-ekview-g3 -- external key viewer

Synopsis

ssh-ekview-g3 [options...] provider

Description

The ssh-ekview-g3 program allows you to export certificates from external key providers. You can further study these certificates with ssh-certview-g3.

This is useful when you want to generate, for example, entries for allowing certificate authentication in the ssh-server-config.xml file. You might need to know the subject names on the certificate.

With ssh-ekview-g3, you can export the certificate and get the information you need from the certificates with ssh-certview-g3.

Options

The following options are available:

- `-h`
  Displays a short help.

- `-i info`
  Uses info as the initialization string for the provider.

- `-k`
  Prints the key paths only.

- `-e keypath`
  Exports certificates at keypath to files.

- `-a`
  Exports all found certificates to files.

- `-b base`
  Uses base when printing integers. For example, the decimal 10 is 'a' in base-16.
Appendix C Egrep Syntax

The Tectia tunneling and FTP-SFTP conversion filter rules can be matched to hostname or IP address patterns specified using the `egrep` syntax. In addition, regular expressions can be used in selectors when specifying ranges of values. The egrep syntax is explained in this section.

C.1 Egrep Patterns

The escape character is a backslash (`\`). You can use it to escape meta characters to use them in their plain character form.

In the following examples literal 'E' and 'F' denote any expression, whether a pattern or a character.

```
(     \n      Start a capturing subexpression.  \n    )  \n        End a capturing subexpression.  \n
E|F  \n   Disjunction, match either E or F (inclusive). E is preferred if both match.  \n
E*  \n   Act as Kleene star, match E zero or more times.  \n
E+  \n   Closure, match E one or more times.  \n
E?  \n   Option, match E optionally once.  \n
.  \n   Match any character except for newline characters (`\n`, `\f`, `\r`) and the `NULL` byte.  \n
E{n}  \n   Match E exactly n times.  \n```
E{n,} or E{n,0}
    Match E n or more times.
E{,n} or E{0,n}
    Match E at most n times.
E{n,m}
    Match E no less than n times and no more than m times.
[
    Start a character set, see Section C.3.
$
    Match the empty string at the end of the input or at the end of a line.
^
    Match the empty string at the start of the input or at the beginning of a line.

C.2 Escaped Tokens for Regex Syntax Egrep

\0n..n
    The literal byte with octal value n..n.
\0
    The NULL byte.
\[1-9]..x
    The literal byte with decimal value [1–9]..x.
\xn..n or \0xn..n
    The literal byte with hexadecimal value n..n.
<
    Match the empty string at the beginning of a word.
>
    Match the empty string at the end of a word.
\b
    Match the empty string at a word boundary.
\B
    Match the empty string provided it is not at a word boundary.
\w
    Match a word-constituent character, equivalent to [a-zA-Z0-9-].
\W
    Match a non-word-constituent character.
\a
    Literal alarm character.
\e
    Literal escape character.
\f
    Literal line feed.
\n
    Literal new line, equivalent to C’s \n so it can be more than one character long.
\r
    Literal carriage return.
\t
    Literal tab.

All other escaped characters denote the literal character itself.

C.3 Character Sets For Egrep

A character set starts with ‘[’ and ends at non-escaped ‘]’ that is not part of a POSIX character set specifier and that does not follow immediately after ‘]’.

The following characters have a special meaning and need to be escaped if meant literally:

- (minus sign)
    A range operator, except immediately after ‘[’, where it loses its special meaning.
^
    If immediately after the starting ‘[’, denotes a complement: the whole character set will be complemented. Otherwise literal ‘^’.

[:alnum:]
    Characters for which 'isalnum' returns true.
[:alpha:]
    Characters for which 'isalpha' returns true.
[:cntrl:]
    Characters for which 'iscntrl' returns true.
[:digit:]  
Characters for which 'isdigit' returns true.

[:graph:]  
Characters for which 'isgraph' returns true.

[:lower:]  
Characters for which 'islower' returns true.

[:print:]  
Characters for which 'isprint' returns true.

[:punct:]  
Characters for which 'ispunct' returns true.

[:space:]  
Characters for which 'isspace' returns true.

[:upper:]  
Characters for which 'isupper' returns true.

[:xdigit:]  
Characters for which 'isxdigit' returns true.

**Example:** [:xdigit:]XY is typically equivalent to [0123456789ABCDEFabcdefXY].

It is also possible to include the predefined escaped character sets into a newly defined one, so [:d:] matches digits and whitespace characters.

Also, escape sequences resulting in literals work inside character sets.
Appendix D Audit Messages

This appendix lists the audit messages generated by the Connection Broker.

1000 KEX_failure
Level: warning
Origin: Tectia Server, Connection Broker

The key exchange failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>User's login name (not present for first KEX)</td>
</tr>
<tr>
<td>Algorithm</td>
<td>KEX algorithm name (not present if failure happens before choosing the algorithm)</td>
</tr>
<tr>
<td>Text</td>
<td>Error description</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
</tbody>
</table>

1001 Algorithm_negotiation_failure
Level: warning
Origin: Tectia Server, Connection Broker

Algorithm negotiation failed - there was no common algorithm in the client's and server's lists.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>User's login name (not present for first KEX)</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Algorithm type</td>
</tr>
<tr>
<td>Client algorithms</td>
<td>Client's algorithm list</td>
</tr>
<tr>
<td>Server algorithms</td>
<td>Server's algorithm list</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
</tbody>
</table>

1002 Algorithm_negotiation_success
Level: informational
Origin: Tectia Server, Connection Broker
Algorithm negotiation succeeded.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>User's login name (not present for first KEX)</td>
</tr>
<tr>
<td>Text</td>
<td>Negotiated algorithms</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
</tbody>
</table>

1003 KEX_success

**Level:** informational  
**Origin:** Connection Broker

Key-exchange was successful.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Kex method name.</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier.</td>
</tr>
<tr>
<td>Protocol-session-Id</td>
<td>Protocol session identifier.</td>
</tr>
</tbody>
</table>

1100 Certificate_validation_failure

**Level:** informational  
**Origin:** Tectia Server, Connection Broker

A received certificate failed to validate correctly under any of the configured CAs.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>User's login name (not present for first KEX)</td>
</tr>
<tr>
<td>Text</td>
<td>Resulting search states for all configured CAs.</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
</tbody>
</table>

1101 Certificate_validation_success

**Level:** informational  
**Origin:** Tectia Server, Connection Broker

A received certificate validated correctly under one or more configured CAs.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>User's login name</td>
</tr>
<tr>
<td>CA List</td>
<td>A list of CAs under which the user's certificate validated correctly.</td>
</tr>
</tbody>
</table>
**1110 CM_find_started**  
**Level:** informational  
**Origin:** Tectia Server, Connection Broker

A low-level search was started in the certificate validation subsystem.

Default log facility: normal

**Argument**  
Session-Id

**Description**  
Session identifier

**Argument**  
Ctx  
Search constraints

**Description**  
Search context

**1111 CM_find_finished**  
**Level:** informational  
**Origin:** Tectia Server, Connection Broker

A low-level find operation has finished in the certificate validation subsystem.

Default log facility: normal

**Argument**  
Ctx

**Description**  
Search constraints.

**1112 CM_cert_not_in_search_interval**  
**Level:** informational  
**Origin:** Tectia Server, Connection Broker

The certificate is not valid during the required time period.

Default log facility: normal

**Argument**  
Ctx  
SubjectName  
Text

**Description**  
Context pointer that identifies the search  
Subject name of the certificate  
Error description

**1113 CM_certificate_revoked**  
**Level:** informational  
**Origin:** Tectia Server, Connection Broker

A certificate was found to be revoked.

Default log facility: normal
### 1114 CM_cert_search_constraint_mismatch

**Level:** informational  
**Origin:** Tectia Server, Connection Broker  

The certificate did not satisfy the constraints set for the search.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubjectName</td>
<td>Subject name of the certificate</td>
</tr>
<tr>
<td>Ctx</td>
<td>The context pointer of the search</td>
</tr>
</tbody>
</table>

### 1115 CM_ldap_search_started

**Level:** informational  
**Origin:** Tectia Server, Connection Broker  

An LDAP search for a CRL or a sub-CA is being started.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Search details</td>
</tr>
</tbody>
</table>

### 1116 CM_ldap_search_success

**Level:** informational  
**Origin:** Tectia Server, Connection Broker  

An LDAP search for a CRL or a sub-CA completed successfully.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Search details</td>
</tr>
</tbody>
</table>

### 1117 CM_ldap_search_failure

**Level:** informational  
**Origin:** Tectia Server, Connection Broker  

The attempt to contact an LDAP server was unsuccessful.

Default log facility: normal
1118 CM_http_search_started

Level: informational
Origin: Tectia Server, Connection Broker

The certificate validation subsystem is initiating a search for a CRL or a sub-CA through the HTTP protocol.

Default log facility: normal

1119 CM_http_search_success

Level: informational
Origin: Tectia Server, Connection Broker

An HTTP request for a CRL or a sub-CA completed successfully.

Default log facility: normal

1120 CM_http_search_failure

Level: informational
Origin: Tectia Server, Connection Broker

An HTTP request for a CRL or a sub-CA failed.

Default log facility: normal

1121 CM_crl_added

Level: informational
Origin: Tectia Server, Connection Broker

A new CRL was successfully added to the certificate validation subsystem.

Default log facility: normal

1122 Certificate_end_point_id_check_success
**Level:** informational  
**Origin:** Connection Broker

End point identity check succeeded.

Default log facility: normal

<table>
<thead>
<tr>
<th><strong>Argument</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Host name</td>
</tr>
<tr>
<td>Text</td>
<td>Explanatory message</td>
</tr>
</tbody>
</table>

**1123 Certificate_end_point_id_check_warning**  
**Level:** informational  
**Origin:** Connection Broker

Certificate end point identity check warning.

Default log facility: normal

<table>
<thead>
<tr>
<th><strong>Argument</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Host name</td>
</tr>
<tr>
<td>Text</td>
<td>Warning message</td>
</tr>
</tbody>
</table>

**1124 Certificate_end_point_id_check_failure**  
**Level:** informational  
**Origin:** Connection Broker

Certificate end point identity check failure.

Default log facility: normal

<table>
<thead>
<tr>
<th><strong>Argument</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Host name</td>
</tr>
<tr>
<td>Text</td>
<td>Error message</td>
</tr>
</tbody>
</table>

**1200 Key_store_create**  
**Level:** informational  
**Origin:** Tectia Server, Connection Broker

Key store created.

Default log facility: normal

**1201 Key_store_create_failed**  
**Level:** warning  
**Origin:** Tectia Server, Connection Broker

Key store creation failed.
Default log facility: normal

1202 Key_store_destroy
Level: informational
Origin: Tectia Server, Connection Broker

Key store destroyed.

Default log facility: normal

1204 Key_store_add_provider
Level: informational
Origin: Tectia Server, Connection Broker

Added a provider to the key store.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Provider type</td>
</tr>
</tbody>
</table>

1205 Key_store_add_provider_failed
Level: warning
Origin: Tectia Server, Connection Broker

Adding a provider to the key store failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Provider type</td>
</tr>
<tr>
<td>EK error</td>
<td>Error message</td>
</tr>
</tbody>
</table>

1206 Key_store_remove_provider
Level: informational
Origin: Tectia Server, Connection Broker

Removed a provider from the key store.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init info</td>
<td>Provider name</td>
</tr>
</tbody>
</table>

1208 Key_store_decrypt
Level: informational
Origin: Tectia Server, Connection Broker
A key was used successfully for decryption.

Default log facility: normal

**Argument** | **Description**
--- | ---
Key path | Key path
Fwd path | Fwd path

**1209 Key_store_decrypt_failed**
**Level:** warning
**Origin:** Tectia Server, Connection Broker

A key was used unsuccessfully for decryption.

Default log facility: normal

**Argument** | **Description**
--- | ---
Key path | Key path
Fwd path | Fwd path
Crypto error | Error string

**1210 Key_store_sign**
**Level:** informational
**Origin:** Tectia Server, Connection Broker

A key was used successfully for signing.

Default log facility: normal

**Argument** | **Description**
--- | ---
Key path | Key path
Fwd path | Fwd path

**1211 Key_store_sign_failed**
**Level:** warning
**Origin:** Tectia Server, Connection Broker

A key was used unsuccessfully for signing.

Default log facility: normal

**Argument** | **Description**
--- | ---
Key path | Key path
Fwd path | Fwd path
Crypto error | Error string

**1212 Key_store_sign_digest**
**Level:** informational
Origin: Tectia Server, Connection Broker

A key was used successfully for signing a digest.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key path</td>
<td>Key path</td>
</tr>
<tr>
<td>Fwd path</td>
<td>Fwd path</td>
</tr>
</tbody>
</table>

1213 Key_store_sign_digest_failed
Level: warning
Origin: Tectia Server, Connection Broker

A key was used unsuccessfully for signing a digest.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key path</td>
<td>Key path</td>
</tr>
<tr>
<td>Fwd path</td>
<td>Fwd path</td>
</tr>
<tr>
<td>Crypto error</td>
<td>Error string</td>
</tr>
</tbody>
</table>

1214 Key_store_ek_provider_failure
Level: warning
Origin: Tectia Server, Connection Broker

External key provider failure.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key path</td>
<td>Key path</td>
</tr>
<tr>
<td>Text</td>
<td>Key label</td>
</tr>
<tr>
<td>Text</td>
<td>Error description</td>
</tr>
</tbody>
</table>

1300 Channel_inbound_statistics
Level: informational
Origin: Connection Broker, Tectia Server

Statistics for the inbound side of a channel (traffic arriving from the network)

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>User's login name</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
<tr>
<td>Channel Id</td>
<td>Local channel id</td>
</tr>
</tbody>
</table>
**Description**
Protocol packet count
Average protocol packet payload size

**1301 Channel_outbound_statistics**
**Level:** informational
**Origin:** Connection Broker, Tectia Server

Statistics for the outbound side of a channel (traffic going to the network)
Default log facility: normal

**Argument**
Packet count
Packet size

**Description**
User's login name
Session identifier
Local channel id
Protocol packet count
Average protocol packet payload size
Final size of outbound channel buffer

**3000 Sft_client_start**
**Level:** debug
**Origin:** Tectia Secure File Transfer clients

File transfer client program was started.
Default log facility: user

**Argument**
Local username
Program
Pid
OpID
Status
Connection
File
Text

**Description**
Local user name
Secure file transfer client program
Client process ID
Operation ID
Result (SUCCESS or FAILED)
Target connection
Path to target file
(Optional) error message and/or additional info

**3001 Sftc_create_file**
**Level:** debug
**Origin:** Tectia Secure File Transfer clients

A new file was created.
Default log facility: user

**Argument**
Local username
Program
Pid
OpID
Status
Connection
File
Text

**Description**
Local user name
Secure file transfer client program
Client process ID
Operation ID
Result (SUCCESS or FAILED)
Target connection
Path to target file
(Optional) error message and/or additional info

**3002 Sftc_truncate_file**
Level: debug
Origin: Tectia Secure File Transfer clients

A file was truncated.

Default log facility: user

**Argument**
- Local username
- Program
- Pid
- OpID
- Status
- Connection
- File
- Text

**Description**
- Local user name
- Secure file transfer client program
- Client process ID
- Operation ID
- Result (SUCCESS or FAILED)
- Target connection
- Path to file
- (Optional) error message and/or additional info

3003 Sftc_modify_file_attrs
Level: informational
Origin: Tectia Secure File Transfer clients

A file attribute was modified.

Default log facility: user

**Argument**
- Local username
- Program
- Pid
- OpID
- Status
- Connection
- File
- Text

**Description**
- Local user name
- Secure file transfer client program
- Client process ID
- Operation ID
- Result (SUCCESS or FAILED)
- Target connection
- Path to file
- (Optional) error message and/or additional info

3004 Sftc_delete_file
Level: notice
Origin: Tectia Secure File Transfer clients

A file was deleted.

Default log facility: user

**Argument**
- Local username
- Program

**Description**
- Local user name
- Secure file transfer client program
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>OpID</td>
<td>Operation ID</td>
</tr>
<tr>
<td>Status</td>
<td>Result (SUCCESS or FAILED)</td>
</tr>
<tr>
<td>Connection</td>
<td>Target connection</td>
</tr>
<tr>
<td>File</td>
<td>Path to file</td>
</tr>
<tr>
<td>Text</td>
<td>(Optional) error message and/or additional info</td>
</tr>
</tbody>
</table>

**3005 Sftc_create_dir**

**Level:** debug
**Origin:** Tectia Secure File Transfer clients

A directory was created.

Default log facility: user

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Program</td>
<td>Secure file transfer client program</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>OpID</td>
<td>Operation ID</td>
</tr>
<tr>
<td>Status</td>
<td>Result (SUCCESS or FAILED)</td>
</tr>
<tr>
<td>Connection</td>
<td>Target connection</td>
</tr>
<tr>
<td>Dir</td>
<td>Path to directory</td>
</tr>
<tr>
<td>Text</td>
<td>(Optional) error message and/or additional info</td>
</tr>
</tbody>
</table>

**3006 Sftc_remove_dir**

**Level:** notice
**Origin:** Tectia Secure File Transfer clients

A directory was removed.

Default log facility: user

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Program</td>
<td>Secure file transfer client program</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>OpID</td>
<td>Operation ID</td>
</tr>
<tr>
<td>Status</td>
<td>Result (SUCCESS or FAILED)</td>
</tr>
<tr>
<td>Connection</td>
<td>Target connection</td>
</tr>
<tr>
<td>Dir</td>
<td>Path to directory</td>
</tr>
<tr>
<td>Text</td>
<td>(Optional) error message and/or additional info</td>
</tr>
</tbody>
</table>

**3007 Sftc_copy_dir_start**

**Level:** notice
Origin: Tectia Secure File Transfer clients

Copying a directory initiated.

Default log facility: user

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Program</td>
<td>Secure file transfer client program</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>OpID</td>
<td>Operation ID</td>
</tr>
<tr>
<td>Source Connection</td>
<td>Source connection</td>
</tr>
<tr>
<td>From</td>
<td>Path to source directory</td>
</tr>
<tr>
<td>Target Connection</td>
<td>Target connection</td>
</tr>
<tr>
<td>To</td>
<td>Path to target directory</td>
</tr>
<tr>
<td>Text</td>
<td>(Optional) error message and/or additional info</td>
</tr>
</tbody>
</table>

3008 Sftc_copy_dir_finished
Level: notice
Origin: Tectia Secure File Transfer clients

Copying a directory completed.

Default log facility: user

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Program</td>
<td>Secure file transfer client program</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>OpID</td>
<td>Operation ID</td>
</tr>
<tr>
<td>Status</td>
<td>Result (SUCCESS or FAILED)</td>
</tr>
<tr>
<td>Source Connection</td>
<td>Source connection</td>
</tr>
<tr>
<td>From</td>
<td>Path to source directory</td>
</tr>
<tr>
<td>Target Connection</td>
<td>Target connection</td>
</tr>
<tr>
<td>To</td>
<td>Path to target directory</td>
</tr>
<tr>
<td>Duration</td>
<td>Duration of copying the directory</td>
</tr>
<tr>
<td>Files</td>
<td>Number of files that were copied</td>
</tr>
<tr>
<td>Data</td>
<td>Amount of data copied in bytes</td>
</tr>
<tr>
<td>Speed</td>
<td>Copy speed in KiB/s</td>
</tr>
<tr>
<td>Text</td>
<td>(Optional) error message and/or additional info</td>
</tr>
</tbody>
</table>

3009 Sftc_move_dir_start
Level: notice
Origin: Tectia Secure File Transfer clients

Moving a directory initiated.
Default log facility: user

**Argument**
- Local username
- Program
- Pid
- OpID
- Source Connection
- From
- Target Connection
- To
- Text

**Description**
- Local user name
- Secure file transfer client program
- Client process ID
- Operation ID
- Source connection
- Path to source directory
- Target connection
- Path to target directory
- (Optional) error message and/or additional info

### 3010 Sftc_move_dir_finished

**Level:** notice

**Origin:** Tectia Secure File Transfer clients

Moving a directory completed.

Default log facility: user

**Argument**
- Local username
- Program
- Pid
- OpID
- Status
- Source Connection
- From
- Target Connection
- To
- Duration
- Files
- Data
- Speed
- Text

**Description**
- Local user name
- Secure file transfer client program
- Client process ID
- Operation ID
- Result (SUCCESS or FAILED)
- Source connection
- Path to source directory
- Target connection
- Path to target directory
- Duration of moving the directory
- Number of files that were moved
- Amount of data transferred in bytes
- Transfer speed in KiB/s
- (Optional) error message and/or additional info

### 3011 Sftc_copy_file_start

**Level:** informational

**Origin:** Tectia Secure File Transfer clients

Copying a file initiated.

Default log facility: user
3012 Sftc_copy_file_finished

**Level:** notice

**Origin:** Tectia Secure File Transfer clients

Copying a file completed.

Default log facility: user

3013 Sftc_move_file_start

**Level:** informational

**Origin:** Tectia Secure File Transfer clients

Moving a file initiated.

Default log facility: user
3014 Sftc_move_file_finished

**Level:** notice

**Origin:** Tectia Secure File Transfer clients

Moving a file completed.

Default log facility: user

3015 Sftc_rename_file

**Level:** informational

**Origin:** Tectia Secure File Transfer clients

A file was renamed.

Default log facility: user
### 3017 Sft_client_command

**Level:** debug  
**Origin:** Tectia Secure File Transfer clients

File transfer client command.

Default log facility: user

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Result (SUCCESS or FAILED)</td>
</tr>
<tr>
<td>Connection</td>
<td>Target connection</td>
</tr>
<tr>
<td>File</td>
<td>Path to file</td>
</tr>
<tr>
<td>Text</td>
<td>(Optional) error message and/or additional info</td>
</tr>
</tbody>
</table>

### 3018 Sftc_open_dir

**Level:** debug  
**Origin:** Tectia Secure File Transfer clients

A directory was opened.

Default log facility: user

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Program</td>
<td>Secure file transfer client program</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>OpID</td>
<td>Operation ID</td>
</tr>
<tr>
<td>Text</td>
<td>Command given on cmd line or defined in batch file</td>
</tr>
</tbody>
</table>

### 6000 Broker_client_connect

**Level:** informational  
**Origin:** Connection Broker

A client connected to the Broker.

Default log facility: discard
### 6001 Broker_client_connect_failed

**Level:** warning  
**Origin:** Connection Broker

A client attempted to connect unsuccessfully to the Broker.

Default log facility: normal

### 6002 Broker_client_disconnect

**Level:** informational  
**Origin:** Connection Broker

A client disconnected from the Broker.

Default log facility: discard

### 6004 Broker_exec_channel_open

**Level:** informational  
**Origin:** Connection Broker

The Broker opened an exec channel.

Default log facility: discard
6005 Broker_exec_channel_open_failed

**Level:** warning

**Origin:** Connection Broker

The Broker failed to open an exec channel for a client.

Default log facility: normal

6006 Broker_tunnel_open

**Level:** informational

**Origin:** Connection Broker

The Broker opened a tunnel for a client.

Default log facility: discard
### 6007 Broker_tunnel_open_failed

**Level:** warning  
**Origin:** Connection Broker

The Broker failed to open a tunnel for a client.

Default log facility: normal

<table>
<thead>
<tr>
<th><strong>Argument</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Server</td>
<td>Server host</td>
</tr>
<tr>
<td>Server Port</td>
<td>Server port</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Tunnel type</td>
<td>Tunnel type</td>
</tr>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

### 6008 Broker_tunnel_listener_open

**Level:** informational  
**Origin:** Connection Broker

The Broker opened a tunnel listener for a client.

Default log facility: discard

<table>
<thead>
<tr>
<th><strong>Argument</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Server</td>
<td>Server host</td>
</tr>
<tr>
<td>Server Port</td>
<td>Server port</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Listener</td>
<td>Listener host</td>
</tr>
<tr>
<td>Listener Port</td>
<td>Listener port</td>
</tr>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Tunnel type</td>
<td>Tunnel type</td>
</tr>
<tr>
<td>Text</td>
<td>Tunnel listener parameters</td>
</tr>
</tbody>
</table>
### 6009 Broker_tunnel_listener_open_failed

**Level:** warning  
**Origin:** Connection Broker

The Broker failed to open a tunnel listener for a client.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Server</td>
<td>Server host</td>
</tr>
<tr>
<td>Server Port</td>
<td>Server port</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Listener</td>
<td>Listener host</td>
</tr>
<tr>
<td>Listener Port</td>
<td>Listener port</td>
</tr>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Tunnel type</td>
<td>Tunnel type</td>
</tr>
<tr>
<td>Text</td>
<td>Tunnel listener parameters</td>
</tr>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

### 6010 Broker_channel_fd_strip

**Level:** informational  
**Origin:** Connection Broker

The Broker destroyed a channel object (and returned the underlying fd to the client).

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Channel Id</td>
<td>Channel ID</td>
</tr>
<tr>
<td>Text</td>
<td>Channel permanent?</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

### 6011 Broker_channel_fd_strip_failed

**Level:** warning  
**Origin:** Connection Broker
The Broker failed to destroy a channel object (and return the underlying fd to the client).

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Channel Id</td>
<td>Channel ID</td>
</tr>
<tr>
<td>Text</td>
<td>Channel permanent?</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

**6012 Broker_channel_control**

**Level:** informational

**Origin:** Connection Broker

The Broker sent a channel control message.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Channel Id</td>
<td>Channel ID</td>
</tr>
<tr>
<td>Command</td>
<td>Command</td>
</tr>
<tr>
<td>Args</td>
<td>Arguments</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

**6013 Broker_channel_control_failed**

**Level:** warning

**Origin:** Connection Broker

The Broker failed to send a channel control message.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Channel Id</td>
<td>Channel ID</td>
</tr>
<tr>
<td>Command</td>
<td>Command</td>
</tr>
<tr>
<td>Args</td>
<td>Arguments</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
</tbody>
</table>
**6014 Broker_channel_close**

**Level:** informational

**Origin:** Connection Broker

The Broker closed a channel.

Default log facility: discard

**6015 Broker_channel_close_failed**

**Level:** warning

**Origin:** Connection Broker

The Broker failed to close a channel.

Default log facility: normal

**6018 Broker_server_version_request**

**Level:** informational

**Origin:** Connection Broker

The Broker requested (and got) the server version.

Default log facility: discard
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

### 6019 Broker_server_version_request_failed

**Level:** warning  
**Origin:** Connection Broker

The Broker failed to get the server version.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Channel Id</td>
<td>Channel ID</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

### 6020 Broker_channel_process_exit

**Level:** informational  
**Origin:** Connection Broker

Channel process exit request was successful.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Channel Id</td>
<td>Channel ID</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

### 6021 Broker_channel_process_exit_failed

**Level:** warning  
**Origin:** Connection Broker

Channel process exit request failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

**6025 BrokerConnector_license_check_failed**

**Level:** warning  
**Origin:** Connection Broker

Connector license check failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Error message</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session id</td>
</tr>
</tbody>
</table>

**6026 BrokerServer_rekey**

**Level:** notice  
**Origin:** Connection Broker

The Broker requested rekeying and it was successful.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

**6027 BrokerServer_rekey_failed**

**Level:** warning  
**Origin:** Connection Broker

The Broker requested rekeying but it failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process ID</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

**6035 BrokerPublickey_upload**

**Level:** informational  
**Origin:** Connection Broker
Public key is uploaded.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Client name</td>
</tr>
<tr>
<td>Pid</td>
<td>Client process id</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Public key hash</td>
<td>Public key hash</td>
</tr>
<tr>
<td>Server</td>
<td>Server name</td>
</tr>
<tr>
<td>Server Port</td>
<td>Server port</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
<tr>
<td>File name</td>
<td>Public key file name</td>
</tr>
</tbody>
</table>

6100 Broker_starting
Level: notice
Origin: Connection Broker

The Broker is starting.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
</tbody>
</table>

6101 Broker_start_failed
Level: warning
Origin: Connection Broker

Starting the Broker failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Success</td>
<td>Error</td>
</tr>
<tr>
<td>Text</td>
<td>Error code</td>
</tr>
</tbody>
</table>

6102 Broker_running
Level: notice
Origin: Connection Broker

The Broker is running.

Default log facility: normal
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Text</td>
<td>Message text</td>
</tr>
</tbody>
</table>

6104 Broker_stopping

**Level:** notice

**Origin:** Connection Broker

The Broker is stopping.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
</tbody>
</table>

6106 Broker_reconfig_started

**Level:** notice

**Origin:** Connection Broker

Reconfiguration started.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
</tbody>
</table>

6108 Broker_reconfig_finished

**Level:** notice

**Origin:** Connection Broker

Reconfiguration finished.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
</tbody>
</table>

6114 Broker_config_deprecated_element

**Level:** warning

**Origin:** Connection Broker

The Broker config contains a deprecated element.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Success</td>
<td>Error</td>
</tr>
</tbody>
</table>

Error code

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Event description.</td>
</tr>
</tbody>
</table>
### 6200 Broker_tcp_connect

**Level:** informational  
**Origin:** Connection Broker

Broker TCP connection attempt was successful.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Src Port</td>
<td>Source port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local username</td>
</tr>
</tbody>
</table>

### 6201 Broker_tcp_connect_failed

**Level:** warning  
**Origin:** Connection Broker

Broker TCP connection attempt failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local username</td>
</tr>
<tr>
<td>NIO error</td>
<td>NIO error</td>
</tr>
</tbody>
</table>

### 6204 Broker_transport_connect

**Level:** informational  
**Origin:** Connection Broker

A transport was connected through TCP.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote username</td>
</tr>
<tr>
<td>Src Port</td>
<td>Source port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local username</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

### 6206 Broker_transport_gateway_connect

**Level:** informational  
**Origin:** Connection Broker
A transport was connected through a gateway handle.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote username</td>
</tr>
<tr>
<td>Local username</td>
<td>Local username</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

6208 Broker_connection_connect

**Level:** informational

**Origin:** Connection Broker

The Broker got successfully a Secure Shell connection up.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
<tr>
<td>Uses gateway?</td>
<td>Is this going through a gateway handle</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
</tbody>
</table>

6209 Broker_connection_connect_failed

**Level:** warning

**Origin:** Connection Broker

The Broker failed to get a Secure Shell connection up.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
<tr>
<td>Uses gateway?</td>
<td>Is this going through a gateway handle</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session ID</td>
</tr>
<tr>
<td>Text</td>
<td>Error code</td>
</tr>
</tbody>
</table>

6210 Broker_connection_disconnect

**Level:** informational
**Origin**: Connection Broker

A Secure Shell connection initiated by the Broker was disconnected.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local username</td>
<td>Local user</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote username</td>
</tr>
</tbody>
</table>

**6211 Broker_unknown_hostkey_accepted**

**Level**: warning

**Origin**: Connection Broker

* The Broker accepted an unknown hostkey without user interaction * because of configuration.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Key digest</td>
</tr>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
</tbody>
</table>

**6212 Broker_new_hostkey**

**Level**: warning

**Origin**: Connection Broker

* First connection to a server or this server hostkey was never * saved before.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Key digest</td>
</tr>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
</tbody>
</table>

**6213 Broker_hostkey_changed**

**Level**: warning

**Origin**: Connection Broker
* Server hostkey is different than the saved hostkey.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Key digest</td>
</tr>
<tr>
<td>Dst</td>
<td>Destination host</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Local username</td>
<td>Local user name</td>
</tr>
<tr>
<td>Remote username</td>
<td>Remote user name</td>
</tr>
</tbody>
</table>

**6301 Broker_userauth_failure**

**Level:** warning

**Origin:** Connection Broker

User authentication failed.

Default log facility: normal

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
</tbody>
</table>

**6302 Broker_userauth_method_success**

**Level:** informational

**Origin:** Connection Broker

A user authentication method succeeded.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Authentication method</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
</tbody>
</table>

**6303 Broker_userauth_method_failure**

**Level:** warning

**Origin:** Connection Broker

A user authentication method failed.

Default log facility: discard

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Authentication method</td>
</tr>
<tr>
<td>Text</td>
<td>Reason</td>
</tr>
<tr>
<td>Session-Id</td>
<td>Session identifier</td>
</tr>
</tbody>
</table>
6401 Connector_filter_rule

**Level:** informational

**Origin:** Connection Broker

SOCKS proxy not tunneling

Default log facility: discard

<table>
<thead>
<tr>
<th><strong>Argument</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>Connector action</td>
</tr>
<tr>
<td>Dst</td>
<td>Address</td>
</tr>
<tr>
<td>Dst Port</td>
<td>Port</td>
</tr>
</tbody>
</table>
Index

Symbols
$HOME/.ssh2, 21
/opt/tectia/etc, 21
_BPX_BATCH_UMASK, 13
_BPX_SHAREAS, 13
_BPXK_AUTOCVT, 13
_BPXK_JOBLOG, 14
_EDC_ADD_ERRNO2, 14

A
accounts, local, 37
active mode FTP, 111
agent forwarding, 105, 179
application tunneling, 105
audit messages, 343
authentication, 23, 37
certificate, 31, 42
host-based, 45
keyboard-interactive, 46
PAM, 46
password, 18, 35, 46
public-key
server, 17, 24
user, 19, 37, 121
RADIUS, 46
SAF, 33–34, 44
SecurID, 46
authentication methods, 23, 173
authority info access, 32
authorization file, 232
authorized_keys directory, 232
authorized_keys file, 232

B
basic configuration, 21, 153

C
CA certificate, 160
case-sensitivity, 66, 263, 294
certificate authentication
server, 31, 158
user, 42
certificate revocation list (CRL), 34
disabling, 33, 160
distribution point, 32
prefetching, 159
certificate viewer, 334
certificates
enrolling, 43
revoked, 31
validating, 158
certification authority (CA), 31, 158
certification, FIPS 140-2, 157
CertKey, 43
CEX
enabling, 16
channel, 105
characters, valid, 263, 294
checkpoint-restart, 65, 278
checksum, 184
ciphers, 170
client configuration, 21
client tools, 21
clients
CMP enrollment, 323
scpg3, 254
sftpg3, 268
sshg3, 243
CMP enrollment client, 323
codepage, 126
command-line options, 22
command-line tools, 225
components, 13
compression, 177
configuration files, 21, 153
syntax, 212
Connection Broker, 10, 16, 21, 153
debugging, 117
reconfiguring, 17
starting, 17
stopping, 17
connection profiles, 185
controlling file transfer, 72, 129
CRL (certificate revocation list)
  disabling, 33, 160
  distribution point, 32
  prefetching, 159
Crypto Express Card (CEX)
  enabling, 16
  cryptographic library, 157
  customer support, 9

D
  data conversion, 126
  data set access, 66, 126
  DD cards, 66
  debugging
    Connection Broker, 117
    default domain, 158
  Diffie-Hellman key exchange, 25, 31
  digital signature, 37
  direct access storage device (DASD), 68
  disabling CRL, 33, 160
  distributing keys, 46
  DNS rule, 194
  Document Type Definition (DTD), 212
documentation, 7
documentation conventions, 7
doD PKI, 160
dynamic tunnels, 111

E
  editing configuration files, 22
  egrep, 339
    character sets, 341
    escaped tokens, 340
    patterns, 339
  enabling use of IBM Crypto Express Card (CEX), 16
  end-point identity check, 158
  enrolling user certificate, 43
  environment variables, 13, 22
    file transfer clients, 90
    file transfer server, 135
  scp3g, 265
  ssh3g3, 251
  sftp3g, 296
  ssh-broker-config.xml, 156
  ssh-broker-ctl, 238
  ssh-broker-g3, 228
  ssh-keyfetch, 320
  ssh-keygen-g3, 312
  ssh-sft-stage, 307
  ssh-translation-table, 304
  escape sequences, 119
  event log, 197
  exclusive-connection, 179
  exit value, sftp3g batch mode, 285
  exit values
    scp3g, 266
    sftp3g, 299
    ssh3g, 253
eXpired CRL, 160
  external key viewer, 338

F
  fallback to plaintext FTP
    command-line option, 197
    in configuration file, 197
  Federal Information Processing Standard (FIPS), 157
  file access permissions, 165
  file transfer, 61, 64–65, 268
    controlling, 72, 129
  file transfer examples, 96
    interactive, 96
    unattended, 99
  file transfer home, 91, 136
  file transfer profiles, 131
    filename-matched, 133
    named, 132
  filename characters, 263, 294
  filename support, 263, 294
  filename-matched file transfer profiles, 131, 133
  filter, 195
  filter engine, 154, 193
  fingerprint, 25, 309–310
FIPS 140-2 certification, 157
firewall, 33–34
forwarding
  agent, 105, 179
  local, 105
  remote, 112
  X11, 105, 179
FTP active mode, 111
FTP commands, 61
FTP passive mode, 110
FTP-SFTP conversion, 10

G
Generation Data Group (GDG), 67, 128
Generation Data Set (GDS), 67, 128
getting started with Tectia client tools, 13
glob patterns, 291
GSSAPI ticket forwarding, 176

H
hashed host key format, 25
Hexl, 335
HFS files, 65–66, 126, 128
HFS Root, 128
home directory, 91, 136
host key, 29
  checking, 181
  directory, 230–231
  hashed format, 25
  public, 25
  resolving, 29
host key algorithms, 172, 187
host-based authentication, 45
hostkeys directory, 230–231
HTTP proxy URL, 158
HTTP repository, 31

I
IBM Crypto Express Card (CEX)
  enabling, 16
identification file, 39, 43, 229
IdKey, 39
idle timeout, 178
incoming tunnels, 112
IP address family, 185
IPv6 address, 105

J
JCL jobs (managing over SFTP), 55
deleting, 57
fetching, 56
listing, 57
submitting, 55
Job Entry Subsystem (JES), 55

K
keepalive messages, 179
keepalive-interval, 179
KEXs, 171
key distribution, 46
key exchange, 25, 31
key fingerprint, 25, 309–310
key pair, 37
key stores, 161, 168
keyboard-interactive authentication, 46
known_hosts file, 30, 166, 231

L
LDAP servers, 159
library, cryptographic, 157
Lightweight Directory Access Protocol (LDAP), 31
local port forwarding, 105
local tunnels, 105
local user account, 37
logging, 197

M
MACs, 170
man pages, 225
man-in-the-middle attack, 25, 31
migrated data sets, 68
modifying configuration files, 22
MVS, 15
MVS data sets, 65–66, 126, 128
MVS user prefix, 91, 136

N
named file transfer profiles, 131–132
network, 210
notation
 path, 263, 294

O
OCSP responders, 159
Online Certificate Status Protocol (OCSP), 31
OpenSSH authorized_keys file, 232
OpenSSH keys, 42, 125, 168
OpenSSH known_hosts file, 166, 231
outgoing tunnels, 105

P
PAM authentication, 46
passive mode FTP, 110
passphrase, 38
password
 stored in a data set, 36
 stored in a file, 36
password authentication, 18, 35, 46
path notation, 263, 294
PEM encoding, 335
permissions, 37
PKCS #11 keys, 168
PKCS #11 token, 43
PKCS #12, 168
PKCS #7 package, 32
Pluggable Authentication Module (PAM), 46
port forwarding, 105
 local, 105
 remote, 112
 restricting, 105
private key
 user, 38, 44
problem situations, 119
profiles for file transfer, 131
proxy rules, 177
pseudo-volume, 92
public key
 host, 25
 user, 38
public-key authentication, 37
 server, 17, 24
 user, 19, 37, 121
public-key signature algorithms, 174

R
RADIUS authentication, 46
random_seed file, 229
reconfiguring the Connection Broker, 17
regular expressions (regex)
in filenames, 263, 294
 syntax, 339
rekey interval, 173
related documents, 7
remote administration, 53
remote environment, 180
remote port forwarding, 112
remote tunnels, 112
restoring archived data sets, 92
return value, sftpg3 batch mode, 285
return values
 scpfg3, 266
 sftpg3, 299
 sshg3, 253
revoked certificate, 31
RFC 4253, 319
RFC 4716, 320
rule, 195
running client programs, 14

S
SAF authentication
 server, 33–34, 168
 user, 44, 168
sample files, 100
SAMPLIB, 100
SCEP client, 330
scpg3, 10, 14, 64, 254
environment variables, 265
exit values, 266
options, 255
secure application connectivity, 105
secure copy (SCP), 64, 254
secure file transfer, 61
Secure File Transfer Protocol (SFTP), 65, 268
Secure Shell client, 121
Secure Shell version 2, 243
secure system administration, 53
SecurID authentication, 46
server authentication
  host key algorithms, 172, 187
  with certificates, 31
  with public key, 24
  with SAF keys, 33–34
server certificate, 31
SFTP
  checkpoint, 65, 278
  commands, 61
  streaming, 65, 278
  tape data sets, 72
sftp-g3, 10, 14, 65, 268
  commands, 273
  environment variables, 296
  exit values, 299
  options, 269
  startup batch file, 268, 298
signature algorithms, 174
smart card, 43
SOCKS Proxy, 10, 21, 153
SOCKS server, 111
SOCKS server URL, 158
ssh-broker-config.xml, 21, 153
ssh-broker-ctl, 9, 16, 233
  commands, 234
  environment variables, 238
  options, 233
ssh-broker-ctl probe-key, 239
  options, 239
ssh-broker-g3, 9, 16, 227
  environment variables, 228
  options, 228
ssh-certview-g3, 334
ssh-cmpclient-g3, 323
  commands, 324
  examples, 328
  options, 325
ssh-ekview-g3, 338
ssh-keydist-g3, 46, 314
ssh-keyfetch, 318
  environment variables, 320
  examples, 320
  options, 318
ssh-keygen-g3, 19, 32, 38, 308
  environment variables, 312
  examples, 312
  options, 308
ssh-scepclient-g3, 330
  commands, 331
  examples, 333
  options, 331
ssh-sft-stage, 305
  environment variables, 307
  options, 306
ssh-socks-proxy, 227
ssh-socks-proxy-config.xml, 21, 153
ssh-socks-proxy-ctl, 233
ssh-translation-table, 301
  environment variables, 304
  options, 301
ssh-troubleshoot, 241
  commands, 242
  options, 241
SSH2, 243
SSH2 keys, 168
SSH_CIPHER Io THR E AS HOL D, 13
SSH_CRYPTOCARD_MAC_GENERATE, 13
SSH_DEBUG_FMT, 14
ssh_ftadv_config, 21
ssh_known_hosts file, 231
ssh_sftp_batch_file, 268, 298
sshd2, 10
SSHENV, 13
sshg3, 10, 14, 243
commands, 250
environment variables, 251
escape sequences, 250
exit values, 253
options, 244
sshsetenv, 13
STAGE, 137, 139
staging, 137
starting the Connection Broker, 17
stopping the Connection Broker, 17
streaming, 65, 278
strict host key checking, 181
support, 9
system administration, 53
system configuration, 153
system log, 197

T

tape data sets
  SFTP, 72
TCP connection
  keepalive, 179
timeout, 178
technical support, 9
Tectia Client components, 13
Tectia client tools for z/OS, 10
Tectia Server for IBM z/OS, 10
Tectia SSH Assistant, 11
terminology, 9
ticket forwarding, 176
timeout, TCP connection, 178
translation table, 301
troubleshooting, 117
tunneling, 105
  applications, 111
  IPv6, 105
  restricting, 105
  X11, 105, 179
tunnels, 105
  local (outgoing), 105
  remote (incoming), 112

U

uploading public keys, 39
user account
  local, 37
user authentication
  host-based, 45
  with certificates, 42
  with keyboard-interactive, 46
  with password, 18, 35
  with public key, 19, 37, 121
  with SAF keys, 44
user certificate, enrolling, 43
user configuration directory, 164
user identity, 187
user key, 39
user-config-directory, 164
using secure copy, 64
using secure file transfer, 65
USS, 15
USS home directory, 91, 136

V

valid characters, 263, 294
volume serial number, 92

W

wildcard, 264, 291, 294

X

X.509 certificates, 32, 43, 168
X11 forwarding, 105, 179
XML attribute
  allow-relay, 190, 193
  allow-ticket-forwarding, 176
data, 188
default-domain, 158
disable-crls, 160
dll-path, 176
domain-identity-check, 158
domain-relay, 190
file, 188
gateway-profile, 186
XML element

accept-unknown-host-keys, 163
address-family, 185
auth-gssapi, 176
auth-hostbased, 174
auth-keyboard-interactive, 176
auth-password, 174
auth-publickey, 174
auth-server-certificate, 181
auth-server-publickey, 181
authentication-method, 174, 183
authentication-methods, 173, 187
authentication-success-message, 183
cacertificate, 160
cert-validation, 158
checksum, 184
cipher, 170
ciphers, 170, 187
close-window-on-disconnect, 184
compression, 177, 188
crl-prefetch, 159
cryptolib, 157
default-settings, 169
dns, 194
dod-pki, 160
evironment, 208
exclusive-connection, 179, 189
file-access-control, 165
filter, 195
filter-engine, 193
forward, 179
forwards, 179, 189
general, 157
host-key-always-ask, 163
hostbased-default-domain, 177
hostkey, 187
hostkey-algorithm, 172
hostkey-algorithms, 172, 187
identification, 162
identity, 187
idle-timeout, 178, 188
issuer-name, 175
keepalive-interval, 179, 189
kex, 171
kexs, 171, 187
key-selection, 175
key-store, 160, 162, 167
key-stores, 161
known-hosts, 166
ldap-server, 159
local-hostname, 174
local-tunnel, 189
log-events, 197–198
log-target, 197
logging, 197
mac, 170
macs, 170, 187
network, 210
ocsp-responder, 159
password, 191
profile, 185
profiles, 185
protocol-parameters, 165
proxy, 177, 188
public-key, 175
quiet-mode, 184
rekey, 173, 187
remote-environment, 180, 191
remote-tunnel, 190
rule, 195
server-authentication-methods, 191
server-banners, 179, 189
sftpg3-mode, 183
static-tunnels, 192
strict-host-key-checking, 163
tcp-connect-timeout, 178, 188
terminal-bell, 184
terminal-selection, 183
tunnel, 193
tunnels, 189
user-config-directory, 164
user-identities, 187
user-keys, 162