z/OS

IBM Ported Tools for z/OS: OpenSSH User's Guide

SA23-2246-00

z/OS

IBM Ported Tools for z/OS: OpenSSH User's Guide

SA23-2246-00

Note:
Before using this information and the product it supports, read the general information under "Notices" on page 343.

This edition applies to version 1, release 2, modification 0 of IBM Ported Tools for z/OS (product number 5655-M23) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this document

This document presents the information you need to set up and use IBM Ported Tools for z/OS: OpenSSH.

Who should use this document?

This document is for system programmers who run a z/OS system with z/OS UNIX System Services (z/OS UNIX), and for their users who use IBM Ported Tools for z/OS: OpenSSH. On other open systems, some system programmer tasks might be done by an administrator.

This document assumes the readers are familiar with z/OS systems as well as with the information for it and its accompanying products.

Where to find more information

Where necessary, this document references information in other documents about the elements and features of z/OS®. For complete titles and order numbers for all z/OS documents, see *z/OS Information Roadmap*.

Softcopy publications

Softcopy z/OS publications are available for web-browsing and PDF versions of the z/OS publications for viewing or printing using Adobe® Acrobat Reader. Visit the z/OS library at http://www.ibm.com/systems/z/os/zos/bkserv/.

IBM Ported Tools for z/OS home page

The IBM Ported Tools for z/OS home page is located at www.ibm.com/servers/eserver/zseries/zos/unix/port_tools.html. It contains a brief description of the IBM Ported Tools for z/OS product, information on how to order it, and supporting documentation.

To order the IBM Ported Tools for z/OS: OpenSSH product, go to the IBM® ShopzSeries Web site at www14.software.ibm.com/webapp/ShopzSeries/ShopzSeries.jsp. Customers can report problems found with this product through their normal support structure.

Discussion list

A mailing list (discussion list) that is not sponsored by IBM might be helpful to users of OpenSSH. It is at http://www.openssh.org/list.html. It contains instructions on subscribing to the OpenSSH mailing list.

To search through past discussions, go to http://marc.theaimsgroup.com/.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an e-mail to mhvrcfs@us.ibm.com
2. Visit the Contact z/OS Web page at http://www.ibm.com/systems/z/os/zos/webqs.html
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   U.S.A.
4. Fax the comments to us as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address
- Your e-mail address
- Your telephone or fax number
- The publication title and order number:
  IBM Ported Tools for z/OS: OpenSSH User's Guide
  SA23-2246-00
- The topic and page number related to your comment
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM zSeries support Web page at http://www.ibm.com/systems/z/support/

Summary of changes

Summary of changes
for SA23-2246-00
IBM Ported Tools for z/OS: OpenSSH

The information in this document was previously presented in *IBM Ported Tools for z/OS User's Guide*, SA22-7985-06. The Xvfb section is now in *IBM Ported Tools for z/OS: Xvfb User's Guide*, SA23-2216-00.

New information

IBM Ported Tools for z/OS: OpenSSH has been upgraded to these Open Source Software releases, resulting in changes to various commands, messages, and configuration files:
- OpenSSH 5.0p1
- OpenSSL 0.9.8k
- zlib 1.2.3

The following chapters are new for this release:
- Chapter 2, "What's new or changed in Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH," on page 3
- Chapter 4, "Migrating to Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH," on page 13
- Chapter 6, "Security topics when using key rings for key management," on page 53. This chapter includes "Managing key rings and restricting access to them" on page 53
- Chapter 12, "SMF Type 119 records for OpenSSH," on page 167

The following sections are new for this release:
- "What you need to verify before using OpenSSH" on page 21
- "Steps for verifying the prerequisites for using OpenSSH" on page 21
- "Steps for setting up server authentication when keys are stored in key rings" on page 29
- "Limiting file system name space for sftp users" on page 45
- "Setting up OpenSSH to collect SMF records" on page 50
- "Steps for setting up user authentication when keys are stored in key rings" on page 68
- "zos_ssh_config" on page 141
- "zos_user_ssh_config" on page 142
- "zos_sshd_config" on page 158
- "List of past vulnerabilities that affected IBM Ported Tools for z/OS: OpenSSH in Version 1 Release 1" on page 187

Two new graphics have been added:
- Figure 2 on page 37
- Figure 6 on page 74

New terms have been added to the glossary.

Updated information

The following sections have been updated:

- "Steps for creating or editing configuration files" on page 24
- "Steps for setting up server authentication when keys are stored in UNIX files" on page 27
- "Steps for setting up user authentication when keys are stored in UNIX files" on page 66
- "Authentication" on page 118
- "User-generated files" on page 164
- Appendix C, "RFCs and Internet drafts," on page 339

The OpenSSH files are now organized in Chapter 10, "OpenSSH files," on page 129 as follows:

- "OpenSSH client configuration files" on page 129
- "OpenSSH daemon configuration files" on page 144
- "Other OpenSSH files" on page 160

Chapter 14, "OpenSSH vulnerabilities," on page 185 contains new and updated OpenSSH vulnerability information.

Chapter 15, "OpenSSH messages," on page 189 contains new and updated messages.

Information from the following APARs have been added:

- APAR OA12576
- APAR OA13041
- APAR OA13595
- APAR OA16934
- APAR OA20690
- APAR OA23227
- APAR OA24067
- APAR OA24527
- APAR OA24548
- APAR OA25411
- APAR OA25412
- APAR OA25816
- APAR OA26338
- APAR OA26660
- APAR OA26871
- APAR OA27987
- APAR OA29825
- APAR OA32325

The term "internationalization" has been replaced with "globalization". The new term has been added to the glossary.

Deleted information

The chapter "What's new or changed in OpenSSH for 3.8.1p1?" has been deleted because the updates are now part of the OpenSSH 5.0p1 base.

Technical changes or additions to the text and graphics are indicated by a vertical line to the left of the change.

Chapter 1. Introduction to IBM Ported Tools for z/OS: OpenSSH

The OpenSSH licensed program is one of the ported applications provided by IBM Ported Tools for z/OS. The current version, which is Version 1 Release 2, can be installed on z/OS 1.10 and later. Users of the previous release (Version 1 Release 1) must migrate to the new release as described in Chapter 4, "Migrating to Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH," on page 13 before using the information in this book. IBM Ported Tools for z/OS Version 1 Release 2 which includes OpenSSH, is available as FMID HOS1120.

In this document, "OpenSSH" refers to the z/OS implementation of OpenSSH. For the open source documentation, see http://www.openssh.org.

What is OpenSSH?

OpenSSH provides secure encryption for both remote login and file transfer. Some of the utilities that it includes are:

- **ssh**, a z/OS client program for logging into a z/OS shell. It can also be used to log into other platform's UNIX® shells. It is an alternative to **rlogin**.
- **scp** for copying files between networks. It is an alternative to **rcp**.
- **sftp** for file transfers over an encrypted **ssh** transport. It is an interactive file transfer program similar to **ftp**.
- **sshd**, a daemon program for **ssh** that listens for connections from clients. The IBM Ported Tools for z/OS: OpenSSH implementation of **sshd** supports both SSH protocol versions 1 and 2 simultaneously.

  The default **sshd** configuration only runs protocol version 2.

Other basic utilities such as **ssh-add**, **ssh-agent**, **ssh-keysign**, **ssh-keyscan**, **ssh-keygen** and **sftp-server** are also included.

To ensure secure encrypted communications, OpenSSH uses ciphers such as Blowfish and 3DES.

IBM Ported Tools for z/OS: OpenSSH provides the following z/OS extensions:

- System Authorization Facility (SAF) key ring. OpenSSH can be configured to allow OpenSSH keys to be stored in SAF key rings. See "Choosing between UNIX files and key rings" on page 53 for more information.
- Multilevel security. It is a security policy that allows the classification of data and users based on a system of hierarchical security levels combined with a system of non-hierarchical security categories. See "Running the sshd daemon in a multilevel-secure environment" on page 44 for more information.
- System Management Facility (SMF). OpenSSH can be configured to collect SMF Type 119 records for both the client and the server. See "Setting up OpenSSH to collect SMF records" on page 50 for more information.

The Internet Engineering Task Force (http://www.ietf.org/) has a Secure Shell (SECSH) working group whose goal is to update and standardize the popular SSH protocol. For information about OpenSSH compliancy to SECSH RFCs and internet drafts, see Appendix C, "RFCs and Internet drafts," on page 339.

Chapter 2. What's new or changed in Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH

This topic documents changes that were introduced in Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH, which includes OpenSSH 5.0p1, OpenSSL 0.9.8k, and zlib 1.2.3. It includes these sections:

Summary of changes to commands

Table 1 lists commands that were changed in Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH.

Table 1. Summary of changes to commands in V1R2 of IBM Ported Tools for z/OS: OpenSSH

Command	Changes
scp	Some of the keywords for the **-o** option have changed. Table 2 on page 7 has a list of the keyword changes for **ssh_config**. OpenSSH can be configured to collect SMF client and server transfer completion records that are associated with **scp**. See "Setting up OpenSSH to collect SMF records" on page 50 for more information. A new environment variable used during SMF-related processing, _ZOS_SMF_FD, is available; it is intended for internal use only. See Table 3 on page 9 for more information. The **scp** executable is shipped as an APF-authorized program. References: • "scp" on page 77 • "ssh_config" on page 129 • "zos_ssh_config" on page 141 • "zos_user_ssh_config" on page 142

Table 1. Summary of changes to commands in V1R2 of IBM Ported Tools for z/OS: OpenSSH (continued)

Command	Changes
sftp	Some of the keywords for the **-o** option have changed. Table 2 on page 7 has a list of the keyword changes for **ssh_config**. New options were added for **ls**: **-a -f -n -r -S -t** OpenSSH can be configured to collect SMF client transfer completion records that are associated with **sftp**. For more information, see "Setting up OpenSSH to collect SMF records" on page 50. A new environment variable used during SMF-related processing, _ZOS_SMF_FD, is available; it is intended for internal use only. See Table 3 on page 9 for more information. The **sftp** executable is shipped as an APF-authorized program. References: • "sftp" on page 79 • "ssh_config" on page 129 • "zos_ssh_config" on page 141 • "zos_user_ssh_config" on page 142
sftp-server	New options (specified on the Subsystem specification) were added : **-e**, **-f** *log_facility*, **-h**, **-l** *log_level* OpenSSH can be configured to collect SMF server transfer completion records that are associated with **sftp-server**. See "Setting up OpenSSH to collect SMF records" on page 50 for more information. A new environment variable used during SMF-related processing, _ZOS_SMF_FD, is available; it is intended for internal use only. See Table 3 on page 9 for more information. The **sftp-server** executable is shipped as an APF-authorized program. References: • "sftp-server" on page 84 • "sshd_config" on page 144 • "zos_sshd_config" on page 158

Table 1. Summary of changes to commands in V1R2 of IBM Ported Tools for z/OS: OpenSSH (continued)

Command	Changes
ssh	Some of the keywords for the **-o** option have changed. Table 2 on page 7 has a list of the keyword changes for **ssh_config**. A new [*bind_address*] argument was added for the **-D**, **-L**, **-R** option (IPv6 addresses). Two new environment variables, _ZOS_SMF_FD (intended for internal use only) and _ZOS_USER_SSH_CONFIG, are available; see Table 3 on page 9 for more information. Two new ciphers ("arcfour128" and "arcfour256") were added for the **-c** option. A new MAC ("umac64@openssh.com") was added for the **-m** option. Two new configuration files, **zos_ssh_config** and **zos_user_ssh_config**, are available; see "New and changed configuration files" on page 7. New options were added: **-K -M -O -S -w** (The **-K** and **-w** options are not supported on z/OS UNIX.) New escape command-line options were added: **-KR -h** *!command* References: • "ssh" on page 85 • "ssh_config" on page 129 • "zos_ssh_config" on page 141 • "zos_user_ssh_config" on page 142
ssh-add	New environment variables were added; see Table 3 on page 9 for more information. _ZOS_SSH_KEY_RING _ZOS_SSH_KEY_RING_LABEL Reference: • "ssh-add" on page 99
ssh-keygen	New command-line options were added: **-F**, **-H**, **-R** New environment variables have been added; see Table 3 on page 9 for more information. _ZOS_SSH_KEY_RING_LABEL Reference: • "ssh-keygen" on page 105
ssh-keyscan	A new command-line option was added: **-H** Reference: • "ssh-keyscan" on page 111
ssh-rand-helper	A new environment variable, _ZOS_SSH_PRNG_CMDS_TIMEOUT, was added; see Table 3 on page 9 for more information. Reference: • "ssh-rand-helper" on page 115

Table 1. Summary of changes to commands in V1R2 of IBM Ported Tools for z/OS: OpenSSH (continued)

Command	Changes
sshd	Some of the keywords for the **-o** option have changed. Table 2 on page 7 has a list of the keyword changes for **sshd_config**. A new option was added to the `authorized_keys` and `ssh_known_hosts` file formats: *zos-key-ring-label="KeyRingOwner/KeyRingName label"*. A new configuration file, **zos_sshd_config**, is available; see "New and changed configuration files" on page 7. The `authorized_keys` file has new option keywords: • **no-user-rc**, which is documented in "no-user-rc" on page 121. • **tunnel**, which is ignored on z/OS UNIX. Support was added to the `ssh_known_hosts` file format for hashed host names and [*host*]:*port* formatting. Two new environment variables, _ZOS_SMF_FD (intended for internal use only) and _ZOS_SSHD_CONFIG, are available; see Table 3 on page 9 for more information. OpenSSH can be configured to collect SMF login failure records for **sshd** as well as server transfer completion records that are associated with "internal-sftp". See "Setting up OpenSSH to collect SMF records" on page 50 for more information. The **sshd** executable is shipped as an APF-authorized program. References: • "sshd" on page 116 • "sshd_config" on page 144 • "zos_sshd_config" on page 158

New and changed configuration files

Table 2 lists configuration files that were added or changed in Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH.

Table 2. Summary of changes to configuration files in V1R2 of IBM Ported Tools for z/OS: OpenSSH

Configuration file	Changes
ssh_config	New keywords have been added: ControlMaster ControlPath ExitOnForwardFailure HashKnownHosts KbdInteractiveDevices (not supported on z/OS UNIX) KbdInteractiveAuthentication (not supported on z/OS UNIX) LocalCommand PermitLocalCommand SendEnv Tunnel (not supported on z/OS UNIX) TunnelDevice (not supported on z/OS UNIX) Two new ciphers ("arcfour128" and "arcfour256") were added for the Ciphers keyword. A new MAC ("umac64@openssh.com") was added for the MACs keyword. A new [*bind_address*] argument for the DynamicForward, LocalForward, and RemoteForward keywords were added. These keywords have been changed: Ciphers DynamicForward HostbasedAuthentication LocalForward MACs RekeyLimit RemoteForward RhostsRSAAuthentication IdentitiesOnly Identityfile Reference: • "ssh_config" on page 129

Table 2. Summary of changes to configuration files in V1R2 of IBM Ported Tools for z/OS: OpenSSH (continued)

Configuration file	Changes
sshd_config	New keywords have been added: AcceptEnv AddressFamily ChrootDirectory ForceCommand HostbasedUsesNameFromPacketOnly Match MaxAuthTries PermitOpen PermitTunnel (not supported on z/OS UNIX) A new value ("delayed") was added for the Compression keyword. A new value ("clientspecified") was added for the GatewayPorts keyword. A new value ("internal-sftp") was added for the Subsystem keyword. Two new ciphers ("arcfour128" and "arcfour256") were added for the Ciphers keyword. A new MAC ("umac64@openssh.com") was added for the MACs keyword. These keywords have been changed: AllowTcpForwarding ChallengeResponseAuthentication (not supported on z/OS UNIX) Ciphers Compression GatewayPorts HostKey MACs PrintLastLog (not supported on z/OS UNIX) Subsystem Reference: • "sshd_config" on page 144
zos_ssh_config	This new configuration file contains system-wide client configuration data that is specific to the z/OS platform. Reference: • "zos_ssh_config" on page 141
zos_sshd_config	This new configuration file contains daemon configuration data that is specific to the z/OS platform. Reference: • "zos_sshd_config" on page 158
zos_user_ssh_config	This new configuration file contains per-user client configuration data that is specific to the z/OS platform. Reference: • "zos_user_ssh_config" on page 142

New environment variables

Table 3 lists environment variables that are new for Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH.

Table 3. List of new environment variables in V1R2 of IBM Ported Tools for z/OS: OpenSSH

Environment variable	Changes
_ZOS_OPENSSH_DEBUG	This new environment variable contains z/OS-specific debug information. It is only used internally and is not for external specification. Reference: None
_ZOS_OPENSSH_MSGCAT	This new environment variable identifies the OpenSSH message catalog to be used when sending OpenSSH error messages. Reference: • "Setting up the message catalog for IBM Ported Tools for z/OS: OpenSSH" on page 38
_ZOS_SMF_FD	This new environment variable is set to the file descriptor number that is used for interprocess communication during SMF-related processing. It is only used internally and is not for external specification. Reference: None
_ZOS_SSH_PRNG_CMDS_TIMEOUT	This new environment variable specifies the timeout value used by **ssh-rand-helper** when running a command from the `/etc/ssh/ssh_prng_cmds` file. Reference: • "ssh-rand-helper — Gather random numbers for OpenSSH" on page 115
_ZOS_SSHD_CONFIG	This new environment variable specifies the path name of the user-defined z/OS-specific daemon configuration file. References: • "sshd" on page 116 • "zos_sshd_config" on page 158
_ZOS_SSH_KEY_RING	This new environment variable specifies the SAF key ring owner and key ring name to use as input. Reference: • "ssh-add" on page 99
_ZOS_SSH_KEY_RING_LABEL	This new environment variable specifies the SAF key ring owner, key ring name, and certificate label to use as input. References: • "ssh-add" on page 99 • "ssh-keygen" on page 105
_ZOS_USER_SSH_CONFIG	This new environment variable specifies the path name of the z/OS-specific per-user OpenSSH client configuration file. References: • "ssh" on page 85 • "zos_user_ssh_config" on page 142

Summary of changes to SYS1.MACLIB

Table 4 lists members of SYS1.MACLIB that were added in Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH.

Table 4. Summary of changes to SYS1.MACLIB in V1R2 of IBM Ported Tools for z/OS: OpenSSH

Sample	Changes
FOTSMF77	This new member contains assembler mapping macros for OpenSSH SMF Type 119 records. Reference: • Chapter 12, "SMF Type 119 records for OpenSSH," on page 167

Summary of changes to non-configuration files in /samples

Table 5 lists files in the `/samples` directory that were added in Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH.

Table 5. Summary of changes to /samples in V1R2 of IBM Ported Tools for z/OS: OpenSSH

Sample	Changes
ssh_smf.h	This new file contains C mapping macros for OpenSSH SMF Type 119 records. Reference: • Chapter 12, "SMF Type 119 records for OpenSSH," on page 167

Chapter 3. How does IBM Ported Tools for z/OS: OpenSSH differ from the open source version?

This topic describes how IBM Ported Tools for z/OS: OpenSSH differs from the open source version.

What IBM Ported Tools for z/OS: OpenSSH supports

sftp can treat files as binary or text. By default, **sftp** assumes that files are binary. Files transferred between EBCDIC and ASCII platforms are not converted. For file transfers between z/OS and ASCII UNIX platforms, you might need to convert your files (treat them as text). The **sftp** ascii subcommand can be used to transfer files in ASCII between the local host and a remote UNIX host. This subcommand assumes that the file data on the network should be encoded in ISO/IEC 8859-1. The **sftp** binary subcommand can be used to disable this conversion and return to performing binary file transfers.

scp treats files as text. By default, **scp** performs ASCII/EBCDIC conversion on files. For more information about how **scp** performs conversion, see Chapter 7, "Globalization on z/OS systems," on page 55.

ssh, sftp and scp are restricted from running in a 3270 environment. The OpenSSH client (**ssh**) cannot be run from OMVS (which is a 3270 session). **ssh** has been disabled under OMVS because passwords are visible while they are being typed by the user in some situations. **sftp** and **scp** invoke **ssh** as part of their processing, so they have the same restriction.

IBM Ported Tools for z/OS: OpenSSH has different default settings. IBM Ported Tools for z/OS: OpenSSH has different default settings than the open source level of OpenSSH. If you share OpenSSH configuration files among platforms, then you should be aware of these differences. The differences are:

- The daemon configuration (**sshd_config**) file has both the AllowTcpForwarding keyword and the Compression keyword set to "no".
- Both the client configuration (**ssh_config**) and the daemon configuration (**sshd_config**) files have the RhostsAuthentication keyword set to "no".
- The daemon configuration (**sshd_config**) file has the Protocol keyword set to 2 as the default setting, which specifies that only protocol version 2 connections are allowed.
- The client configuration (**ssh_config**) file has the Protocol keyword set to 2, which specifies that only protocol version 2 connections are allowed.
- The default locations of z/OS executables might differ than on other platforms, so the Subsystem specification of **sftp** might contain a different path on z/OS. On z/OS it is set to:

```
Subsystem     sftp    /usr/lib/ssh/sftp-server
```

IBM Ported Tools for z/OS: OpenSSH provides support that is unique to z/OS. The following z/OS extensions are provided:

- System Authorization Facility (SAF) key ring. OpenSSH can be configured to allow OpenSSH keys to be stored in SAF key rings. See "Choosing between UNIX files and key rings" on page 53 for more information.

- Multilevel security. It is a security policy that allows the classification of data and users based on a system of hierarchical security levels combined with a system of non-hierarchical security categories. See "Running the sshd daemon in a multilevel-secure environment" on page 44.
- System Management Facility (SMF). OpenSSH can be configured to collect SMF Type 119 records for both the client and the server. See "Setting up OpenSSH to collect SMF records" on page 50 for more information.

What IBM Ported Tools for z/OS: OpenSSH does not support

IBM Ported Tools for z/OS: OpenSSH does not support the following functionality:

- AFS® token passing
- Kerberos
- Pluggable Authentication Module (PAM)
- Print last log
- GSS-API
- Smart cards
- "Keyboard-interactive" user authentication
- TCP wrappers
- Tunnel device forwarding

User-defined subsystems treat data as binary. Subsystems are a feature of SSH protocol version 2 which facilitate the use of **ssh** as a secure transport for other applications such as **sftp**. However, you can define your own subsystem using the Subsystem keyword of **sshd_config**. The subsystem is then invoked as a remote command. For example:

```
Subsystem  backups   /home/billyjc/backups.sh
```

Because network data for a subsystem is treated as binary, any output generated by a subsystem will not be displayed correctly between z/OS systems unless steps are taken to convert the data.

IBM Ported Tools for z/OS: OpenSSH does not support multibyte locales. IBM Ported Tools for z/OS: OpenSSH does not support running in multibyte locales. It currently only supports single-byte locales that are compatible with ASCII coded character set ISO/IEC 8859-1. For more information, see Chapter 7, "Globalization on z/OS systems," on page 55.

Chapter 4. Migrating to Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH

This information assumes that you are migrating from Version 1 Release 1 of IBM Ported Tools for z/OS: OpenSSH and that it has been upgraded to OpenSSH 3.8.1p1 with all the available PTFs applied.

If you are currently using Version 1 Release 1 of IBM Ported Tools for z/OS: OpenSSH and the OpenSSH level is 3.5p1, you must migrate to 3.8.1p1 first before you can migrate to IBM Ported Tools for z/OS: OpenSSH Version 1 Release 2, which uses the 5.0p1 level of OpenSSH. For information about migrating to 3.8.1p1, refer to *IBM Ported Tools for z/OS User's Guide*, SA22-7985.

If you are migrating from an unsupported version, you must first migrate to IBM Ported Tools for z/OS: OpenSSH Version 1 Release 1 that has been upgraded to OpenSSH 3.8.1p1 before migrating to IBM Ported Tools for z/OS: OpenSSH Version 1 Release 2.

Considerations when migrating from IBM Ported Tools for z/OS: OpenSSH Version 1 Release 1

This section describes coexistence and compatibility considerations when migrating from IBM Ported Tools for z/OS: OpenSSH Version 1 Release 1.

Coexistence considerations

In a sysplex environment, some systems might share the same configuration. They might also share the `ssh_known_hosts` or `authorized_keys` files. However, those systems might have different versions of **ssh** or **sshd**. In that situation, the previous version of the command might exit with an error message because it does not support the new features. For a list of the configuration keywords that were introduced in IBM Ported Tools for z/OS: OpenSSH Version 1 Release 2, see Table 2 on page 7. For a list of the `ssh_known_hosts` or `authorized_keys` files options that were introduced, see "Summary of changes to commands" on page 3.

Tips:

- To avoid sharing the same configuration file, the user can specify the local configuration file using '`-F config_file`' for **ssh** and '`-f config_file`' for **sshd** on the command line.
- To avoid sharing the same `ssh_known_hosts` file, the user can specify the local file using the **ssh_config** GlobalKnownHostsFile or UserKnownHostsFile keywords.
- To avoid sharing the same `authorized_keys` file, the user can specify the local file using the **sshd_config** AuthorizedKeysFile keyword.

Compatibility considerations

When a newer version of the SSH client is trying to connect to a previous version of the **sshd** daemon, connection might not be established due to incompatibility of the new configuration options. For a list of the configuration keywords that were introduced in IBM Ported Tools for z/OS: OpenSSH Version 1 Release 2, see Table 2 on page 7.

Migration actions for IBM Ported Tools for z/OS: OpenSSH Version 1 Release 2

Migration to IBM Ported Tools for z/OS: OpenSSH Version 1 Release 2 might require certain actions, which are listed as follows:

Setting up the XPLINK environment for use by IBM Ported Tools for z/OS: OpenSSH

Description: Beginning in Version 1 Release 2, IBM Ported Tools for z/OS: OpenSSH is an XPLINK application. XPLINK (Extra Performance Linkage) is a type of call linkage that can improve performance in an environment of frequent calls between small functions.

Is the migration action required?	Yes, to ensure optimal performance.

Steps to take: To set up the XPLINK environment (that is, to initialize the resources necessary to run an XPLINK application), do the following:

- Put the Language Environment® run-time library SCEERUN2 in the LNKLST member of SYS1.PARMLIB.
- Put the XPLINK modules in SCEERUN2 in the dynamic LPA.

Reference information:

- For more information about XPLINK, see *z/OS Language Environment Programming Guide*.
- For more information about placing SCEERUN2 in LNKLST, see *z/OS Language Environment Customization*.
- For more information about LNKLST, see *z/OS MVS Initialization and Tuning Reference*.

Changes to the sftp command that might require a migration action

Table 6 on page 15 lists the changes to the **sftp** command that might require a migration action and the accompanying actions.

Table 6. Changes to the sftp command that might require a migration action

What changed	Migration action needed?
-b option When the **sftp** command is run with the **-b** option, the **-oBatchMode=yes** argument is now passed to the **ssh** command. For more information, see "-b option" on page 80.	Yes, if you use the **sftp** command with the **-b** option and require password, passphrase or host key prompts during authentication. **Action:** Run the **sftp** command with **-oBatchMode=no** as the first option.

Changes to the ssh command that might require a migration action

Table 7 lists the changes to the **ssh** command that might require a migration action and the accompanying actions.

Table 7. Changes to the ssh command that might require a migration action

What changed	Migration action needed?
Previously, if the user was using the default configuration file (`~/.ssh/config`), the owner or permissions on the file was not checked. Now **ssh** issues an error message and exits if the file is not owned by the user or if the file is writable by the world or the file's group.	Yes, if your file has incorrect owner or permissions. More information about the requirements for those can be found in Table 19 on page 164. **Action:** Correct the settings so they adhere to the new requirements.
-c option Previously, the default cipher list did not contain arcfour128 and arcfour256. Now the default cipher list contains arcfour128 and arcfour256. The order was also changed to prefer ciphers that are not susceptible to security vulnerability CVE-2008-5161. Most customers will not be affected by the changed default. The complete list of ciphers used by **ssh** can be found in **ssh_config** (see "Ciphers" on page 130).	Yes, if you use the previous default list and do not want to allow the new ciphers or the new order of the preferred ciphers. The previous default list was the following: aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc,aes128-ctr,aes192-ctr,aes256-ctr. (Typically the ciphers are one long unbroken line; in the preceding example, the ciphers are not shown as one unbroken line due to space limitations.) **Action:** Specify the previous default list.
-L, **-R** options Previously, addresses containing a colon (:) character could be parsed using the forward slash (/) character and vice versa. Now addresses containing delimiter characters (: or /) must be enclosed in square brackets. For more information, see: • "-L option" on page 89 • "-R option" on page 90	Yes, if you use an address that contains delimiter characters. **Action:** Enclose the address in square brackets.
-m option Previously, the default MACs list did not contain umac64@openssh.com. Now the default MACs list contains umac64@openssh.com. Most customers will not be affected by the changed default. The complete list of MACs used by **ssh** can be found in **ssh_config** (see "MACs" on page 136).	Yes, if you use the previous default list and do not want to allow the new MAC. The previous default list was the following: hmac-md5,hmac-sha1,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96. (Typically the MACs are one long unbroken line; in the preceding example, the MACs are not shown as one unbroken line due to space limitations.) **Action:** Specify the previous default list.

Table 7. Changes to the ssh command that might require a migration action (continued)

What changed	Migration action needed?
-o option Some of the keywords have had changes.	Yes, if you use one of the keywords that has changed. For a list of the keywords that have changed and corresponding migration actions (if any), see "Changes to the ssh_config file that might require a migration action."

Changes to the ssh_config file that might require a migration action

Table 8 lists the changes to the **ssh_config** file that might require a migration action and the accompanying actions.

Table 8. Changes to the ssh_config file that might require a migration action

What changed	Migration action needed?
Ciphers Previously, the default cipher list did not contain arcfour128 and arcfour256. Now the default cipher list contains arcfour128 and arcfour256. The order was also changed to prefer ciphers that are not susceptible to security vulnerability CVE-2008-5161. Most customers will not be affected by the changed default. The complete list of ciphers can be found in **ssh_config** (see "Ciphers" on page 130).	Yes, if you use the previous default list and do not want to allow the new ciphers or the new order of the preferred ciphers. The previous default list was the following: aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc,aes128-ctr,aes192-ctr,aes256-ctr. (Typically the ciphers are one long unbroken line; in the preceding example, the ciphers are not shown as one unbroken line due to space limitations.) **Action:** Specify the previous default list.
LocalForward, **RemoteForward** Previously, addresses containing a colon (:) character could be parsed using the forward slash (/) character and vice versa. Now addresses containing delimiter characters (: or /) must be enclosed in square brackets. For more information, see: • "LocalForward" on page 136 • "RemoteForward" on page 137	Yes, if you use an address that contains delimiter characters. **Action:** Enclose the address in square brackets.
MACs Previously, the default MACs list did not contain umac64@openssh.com. Now the default MACs list contains umac64@openssh.com. Most customers will not be affected by the changed default. The complete list of MACs can be found in **ssh_config** (see "MACs" on page 136).	Yes, if you use the previous default list and do not want to allow the new MAC. The previous default list was the following: hmac-md5,hmac-sha1,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96. (Typically the MACs are one long unbroken line; in the preceding example, the MACs are not shown as one unbroken line due to space limitations.) Specify the previous default list.
ProxyCommand Instead of running ProxyCommand with `/bin/sh`, the user's shell as set in the SHELL environment variable is used.	Yes, if you use a shell other than `/bin/sh` (for example, tcsh). **Action:** Make sure that ProxyCommand conforms to your shell's syntax. The description of the **ssh_config** keyword "ProxyCommand" on page 137 has more information about specifying the command to connect to the server.

Table 8. Changes to the ssh_config file that might require a migration action (continued)

What changed	Migration action needed?
RekeyLimit Previously, the minimum value was 0. Now the minimum value is 16.	Yes, if you use a RekeyLimit value that is less than 16. **Action:** Change the value so that the RekeyLimit value is greater than or equal to 16. The description of the **ssh_config** keyword "RekeyLimit" on page 137 has more information about specifying the values.

Changes to the sshd command that might require a migration action

Table 9 lists the changes to the **sshd** command that might require a migration action and the accompanying actions.

Table 9. Changes to the sshd command that might require a migration action

What changed	Migration action needed?
Previously, the **sshd** daemon could be started using a relative path name (for example, `./sshd`). Now a full path name must be used instead of the relative path name.	Yes, if you use a relative path name when starting the **sshd** daemon. Otherwise, **sshd** issues an error message and exits. **Action:** Change the startup process to use the full path name instead of a relative path name.
permitopen authorized keys file option Previously, addresses containing a colon (:) character could be parsed using the forward slash (/) character and vice versa. Now addresses containing delimiter characters (: or /) must be enclosed in square brackets. "permitopen" on page 121 describes the file option in more detail.	Yes, if you use an address that contains delimiter characters. **Action:** Enclose the address in square brackets.
-o option Some of the keywords have had changes.	Yes, if you use one of the keywords that has changed. For a list of the keywords that have changed and corresponding migration actions (if any), see "Changes to the sshd_config file that might require a migration action."

Changes to the sshd_config file that might require a migration action

Table 10 lists the changes to the **sshd_config** file that might require a migration action and the accompanying actions.

Table 10. Changes to the sshd_config file that might require a migration action

What changed	Migration action needed?
AllowTCPForwarding Previously, the default value was "yes". Now it is "no".	Yes, if you want to continue to allow port forwarding. This default was changed to reduce exposure to a vulnerability reported as CVE-2004-1653. The keyword is described in "AllowTcpForwarding" on page 145. **Action:** Set AllowTCPForwarding to "yes".

Table 10. Changes to the sshd_config file that might require a migration action (continued)

What changed	Migration action needed?
ChallengeResponseAuthentication Previously, the default value was "yes". Now it is "no". The keyword is described in "ChallengeResponseAuthentication" on page 146.	No, because ChallengeResponseAuthentication is not supported on z/OS systems.
Ciphers Previously, the default cipher list did not contain arcfour128 and arcfour256. Now the default cipher list contains arcfour128 and arcfour256. The order was also changed to prefer ciphers that are not susceptible to security vulnerability CVE-2008-5161. Most customers will not be affected by the changed default. The complete list of ciphers used by **sshd** can be found in **sshd_config** (see"Ciphers" on page 147).	Yes, if you use the previous default list and do not want to allow the new ciphers or the new order of the preferred ciphers. The previous default list was the following: aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc,aes128-ctr,aes192-ctr,aes256-ctr. The previous default list was the following: (Typically the ciphers are one long unbroken line; in the preceding example, the ciphers are not shown as one unbroken line due to space limitations.) **Action:** Specify the previous default list.
MACs Previously, the default MACs list did not contain umac64@openssh.com. Now the default MACs list contains umac64@openssh.com. Most customers will not be affected by the changed default. The complete list of MACs used by **sshd** can be found in **sshd_config** (see "MACs" on page 152).	Yes, if you use the previous default list and do not want to allow the new MAC. The previous default list was the following: hmac-md5,hmac-sha1,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96. (Typically the MACs are one long unbroken line; in the preceding example, the MACs are not shown as one unbroken line due to space limitations.) Specify the previous default list.
PrintLastLog Previously, the default value was "yes". Now it is "no". The keyword is described in "PrintLastLog" on page 155.	No, because PrintLastLog is not supported on z/OS systems.

Changes to the ssh-keygen command that might require a migration action

Table 11 lists the changes to the **ssh-keygen** command that might require a migration action and the accompanying actions.

Table 11. Changes to the ssh-keygen command that might require a migration action

What changed	Migration action needed?
-b option (for RSA) Previously, the minimum RSA key size on the **ssh-keygen -b** option was 512 bits and the default was 1024 bits. Now the minimum RSA key size is 768 bits and the default is 2048 bits. The maximum remains 32768 bits. For more information, see "-b option" on page 107.	Yes, if you are using **ssh-keygen** to generate RSA keys with a size that is less than 768 bits. **Action:** Use **ssh-keygen** to generate new RSA keys based on the new minimum size. If improved security is desired, regenerate existing RSA keys if their size is less than 768 bits.
-b option (for DSA) Previously, the DSA key size on the **ssh-keygen -b** option was allowed to be between 512 and 32768 bits. Now the DSA key size must be 1024 bits. For more information, see "-b option" on page 107.	Yes, if you are using **ssh-keygen** to generate DSA keys with a size that is not equal to 1024 bits. **Action:** Use **ssh-keygen** to generate new DSA keys based on the new size requirement. If FIPS 186-2 compliance is required, regenerate existing DSA keys if their size is not 1024 bits.

Table 11. Changes to the ssh-keygen command that might require a migration action (continued)

What changed	Migration action needed?
-f option Instead of truncating a long file name at 1023 characters, a message is issued. For more information, see "-f option" on page 107.	No, because long file names will continue to be invalid.
-r option Previously, if the file name was not specified, a prompt for the file name was issued. Now the default file names for RSA and DSA keys are used instead. For more information, see "-r option" on page 109.	Yes, if you did not specify a file name. **Action:** Specify the file name on the **ssh-keygen** command.
ssh-keygen without the **-d** or **-t** options Previously, if **ssh-keygen** was issued without the **-d** or **-t** options, a message was issued. Now RSA is used as the default key type. For more information, see "-d option" on page 107 and "-t option" on page 109.	No, because previously successful **ssh-keygen** commands will continue to be successful.

Preventing message numbers from being associated with OpenSSH error messages

Description: Previously, to associate message numbers (for example, FOTSnnnn) with OpenSSH error messages, the NLSPATH environment variable had to include the following path: `/usr/lib/nls/msg/%L/%N.cat`. Starting in Version 1 Release 2, message numbers for IBM Ported Tools for z/OS: OpenSSH are associated with OpenSSH error messages by default.

Is the migration action required?	Yes, if you do not want message numbers to be associated with OpenSSH error messages.

Steps to take: If you do not want message numbers to be associated with OpenSSH error messages, then set environment variable _ZOS_OPENSSH_MSGCAT="NONE" before running an OpenSSH command. If you have previously modified the NLSPATH environment variable, you do not need to make any changes to it.

Reference information: For more information, see "Setting up the message catalog for IBM Ported Tools for z/OS: OpenSSH" on page 38.

Chapter 5. For system administrators

This topic describes the various tasks that the system administrator handles.

Rule: All files used by IBM Ported Tools for z/OS: OpenSSH (such as key files and configuration files) must be in the IBM-1047 code set, except for the **rc** files (`/etc/ssh/sshrc` and `~/.ssh/rc`). Those files are parsed by `/bin/sh` and should be in the code set of the current locale. Do not use the `/etc/ssh/sshrc` file if there is a possibility of the users on the system running in different locales.

Restriction: IBM Ported Tools for z/OS: OpenSSH does not run in multibyte locales.

Differences between sftp and FTP

OpenSSH's **sftp** and IBM Communications Server's FTP with System SSL differ from each other. OpenSSH's **sftp** is an Open Source implementation of the IETF Secure Shell (SECSH) "SSH File Transfer Protocol " Internet Draft. OpenSSH uses a statically linked OpenSSL archive library to perform its cryptographic functions. OpenSSH provides some key management facilities with the **ssh-keygen** command. However, this support is not integrated with System SSL support provided by IBM. OpenSSH uses the security product when performing password authentication and when extracting keys from certificates associated with SAF key rings. The public key authentication processing itself is overseen by the OpenSSH daemon.

For information about the IETF SECSH internet drafts, see Appendix C, "RFCs and Internet drafts," on page 339.

The Communications Server FTP server and client support Transport Layer Security (TLS). The FTP client and server negotiate the use of TLS based on a subset of the FTP security negotiation functions documented in RFC 2228. FTP uses z/OS System SSL, and therefore can use the cryptographic hardware. For more information about FTP, see *z/OS Communications Server: IP Configuration Guide*.

Because **sftp** and FTP with System SSL do not use the same protocol, they cannot communicate with each other to establish a secure session.

Restriction: OpenSSH's **sftp** support does not include built-in support for MVS™ data sets. For alternate ways to access MVS data sets within **sftp**, see Appendix A, "Accessing MVS data sets within sftp," on page 333.

What you need to verify before using OpenSSH

Before using OpenSSH, the system administrator should check that all prerequisites have been met.

Steps for verifying the prerequisites for using OpenSSH

Before you begin: Perform the following steps to verify that the prerequisites for using OpenSSH have been met.

1. Using Table 12 on page 22 as a reference, check that certain directories were set up correctly when IBM Ported Tools for z/OS: OpenSSH was installed.

Table 12. List of directories and needed permissions

Directory	Permission	Owner	Notes
/var/empty	755	UID(0)	Must be empty. It is used as the home directory for the SSHD (unprivileged) user. For more information about privilege separation, see "Step for creating the sshd privilege separation user" on page 38.
/var/run	755	UID(0)	Holds the `sshd.pid` file, which contains the process ID of the most recently started OpenSSH daemon. If another directory is preferred, the PidFile configuration option can be specified in the daemon's **sshd_config** file. For more information, see "sshd_config" on page 144. Also holds the `sshd.mm.XXXXXXXX` temporary files which are used for compression with privilege separation.
/etc/ssh	755	UID(0)	Holds the configuration files for **ssh** and **sshd**.

2. If running on z/OS 1.10 or z/OS 1.11, check that the PTFs for the following APARs have been applied:
   - PK86329
   - OA29401

3. Check that the **sshd** daemon has been installed with the program control, APF-authorized, and noshareas extended attributes. To verify that these extended attributes have been set properly, issue the following shell command:

   ```
   ls -El /usr/sbin/sshd
   ```

   The output should be similar to the following:

   ```
   -rwxr--r-- ap-- 2 SYSADM 1 5783552 Jul 9 08:24 /usr/sbin/sshd
   ```

   The 'p' indicates that the program control extended attribute is set. The 'a' indicates that the APF-authorized extended attribute is set. The lack of an 's' after the 'p' indicates that the noshareas extended attribute is set. If the output is not correct, then you must set the attributes as follows.
   - To set the noshareas extended attribute, issue the following shell command:

     ```
     extattr -s /usr/sbin/sshd
     ```

   - If you are a UID(0) user with at least READ access to the BPX.FILEATTR.PROGCTL resource in the FACILITY class, you can set the program control extended attribute by issuing the following shell command:

     ```
     extattr +p /usr/sbin/sshd
     ```

   - If you are a UID(0) user with at least READ access to the BPX.FILEATTR.APF resource in the FACILITY class, you can set the APF-authorized extended attribute by issuing the following shell command:

     ```
     extattr +a /usr/sbin/sshd
     ```

   In addition, you might also need to ensure that the Language Environment run-time library is defined to program control, as shown in the following example:

```
SETROPTS WHEN(PROGRAM)
RDEFINE PROGRAM * ADDMEM
('CEE.SCEERUN'/volser/NOPADCHK
'SYS1.LINKLIB'/'******'/NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

4. Check that the **scp**, **sftp**, and **sftp-server** programs have been installed with the APF-authorized attribute. To verify that this extended attribute is set properly, issue the following shell command for each program:

   ```
   ls -El progname
   ```

   where *progname* is `/bin/scp`, `/bin/sftp`, or `/usr/lib/ssh/sftp-server`.

   The output should be similar to the following:

   ```
   -rwxr-xr-x a-s- 2 SYSADM 1 5783552 Jul 9 08:24 progname
   ```

   The 'a' indicates that the APF-authorized extended attribute is set. If the output is not correct, then you must set the attribute as follows.

   - If you are UID(0) user with at least READ access to the BPX.FILEATTR.APF resource in the FACILITY class, you can set the APF-authorized extended attribute by issuing the following shell command:

     ```
     extattr +a progname
     ```

5. Check that the **ssh** and **ssh-keysign** programs have been installed with the noshareas extended attribute. To verify that this extended attribute is set properly, issue the following shell command for each program:

   ```
   ls -El progname
   ```

   where progname is `/bin/ssh` or `/usr/lib/ssh/ssh-keysign`. The output should be similar to the following:

   ```
   -rwxr-xr-x ---- 2 SYSADM 1 5783552 Jul 9 08:24 progname
   ```

   The third - in '----' indicates that the noshareas extended attribute is set. If the output is not correct, then you must set the noshareas extended attribute. For example, to set the noshareas extended attribute for `/bin/ssh`, issue the following shell command:

   ```
   extattr -s /bin/ssh
   ```

When you are done, you have verified that the prerequisites for using OpenSSH have been met.

For more information about program control, see *z/OS UNIX System Services Planning*.

Setting up the sshd daemon

Before the system administrator can start the **sshd** daemon, the following setup tasks must be done:

- The configuration files must be created or edited, as described in "Steps for creating or editing configuration files" on page 24.

- Server authentication must be set up as described in "Steps for setting up server authentication when keys are stored in UNIX files" on page 27 and "Steps for setting up server authentication when keys are stored in key rings" on page 29.
- The **sshd** privilege separation user must be created as described in "Step for creating the sshd privilege separation user" on page 38.

Setting up the message catalog for IBM Ported Tools for z/OS: OpenSSH is an optional task. The task is described in "Setting up the message catalog for IBM Ported Tools for z/OS: OpenSSH" on page 38.

Steps for creating or editing configuration files

Perform the following steps to create or edit the configuration files.

1. Copy the configuration files from the `/samples` directory to the `/etc/ssh` directory. Store them in the IBM-1047 (EBCDIC) code set. Additionally, set the appropriate mode for some of the copied files.

```
cp -p /samples/sshd_config      /etc/ssh/sshd_config
cp -p /samples/ssh_config       /etc/ssh/ssh_config
cp -p /samples/moduli           /etc/ssh/moduli
cp -p /samples/ssh_prng_cmds    /etc/ssh/ssh_prng_cmds
cp -p /samples/zos_sshd_config  /etc/ssh/zos_sshd_config
cp -p /samples/zos_ssh_config   /etc/ssh/zos_ssh_config
chmod 600 /etc/ssh/sshd_config
chmod 600 /etc/ssh/zos_sshd_config
```

   "Configuration files" on page 163 lists the permission and UID settings for each configuration file.

---

2. Modify the `/etc/ssh/sshd_config` file to control the SSH server's authentication methods allowed, protocols, and ciphers supported, port forwarding, and session control options. For more details, see "sshd" on page 116 and "sshd_config" on page 144.

   Appendix B, "OpenSSH - port forwarding examples," on page 335 has examples of port forwarding.

---

3. Modify the `/etc/ssh/ssh_config` file to control the SSH client-side authentication methods, protocols, ciphers, port forwarding settings and session control options. For more details, see "ssh" on page 85 and "ssh_config" on page 129.

   Notes:

   a. The settings in this configuration file provide system defaults. They can be overridden by the user's **ssh** configuration in `~/.ssh/config` file or by command-line options.

   b. The **ssh_config** file can be shared across multiple systems with client configuration options that are tailored to the specific local system being used. To share the file, preface groups of configuration options with the Host keyword.

---

4. Configure the TCP port. By default, **sshd** listens on TCP port 22. Because this is in the range of ports numbered 1–1023, it is considered to be a privileged TCP port. Only daemons running as a superuser are allowed to listen on these ports unless TCP is configured to unrestrict low ports.

   You can configure **sshd** to listen on a different port with the Port keyword or the **-p** command-line option (see "sshd_config" on page 144).

Example: An example of an **sshd_config** entry is:

```
Port 1022
```

If you want to reserve the port for **sshd** daemon use, add the following lines to PROFILE.TCPIP within the Port statements:

```
PORT
22 TCP SSHD* ; port for sshd daemon
```

The job name must have the wildcard format of SSHD* because as the sshd daemon starts, it creates child tasks starting with SSHDn where *n* is a number between 1 and 9. Depending on your system, the resulting daemon task will be one of these child tasks so a D OMVS,A=ALL will show SSHDn as the daemon task. Use of this wildcard means that TCP/IP cannot automatically restart the daemon if it goes down. See "Starting the sshd daemon" on page 39 for information about starting the OpenSSH daemon.

---

5. Set up random number generation. You have two choices.

   - You can use **ssh-rand-helper** to gather random numbers. The sample file copied into the `/etc/ssh/ssh_prng_cmds` file, which is used by **ssh-rand-helper** to gather random numbers of cryptographic quality, should provide enough entropy for most installations. To produce random numbers, the OpenSSH entropy collector runs the commands listed in this file and adds the output to other sources of entropy. OpenSSH depends on unpredictable random numbers for generating keys, performing digital signatures, and forming cryptographic challenges. For more information about **ssh-rand-helper,** see "ssh-rand-helper" on page 115.

     OpenSSH users might be required to have special authority to successfully run some of the commands listed in the `/etc/ssh/ssh_prng_cmds` file. As a result, you might want to remove these commands from the file to avoid authority failures or you might need to replace these commands to ensure that enough entropy is generated. For example, the SERVAUTH NETSTAT profile controls access to the **netstat** command.

     Rule: ssh-rand-helper must generate at least 48 random bytes to ensure enough entropy is generated for OpenSSH.

     Tip: To provide more randomness, add more commands to the `/etc/ssh/ssh_prng_cmds` file. However, OpenSSH performance might be affected.

   - If Integrated Cryptographic Service Facility (ICSF) is available, you can use hardware support (`/dev/random` or `/dev/urandom`) to generate random numbers. For more information about using hardware support, see "Using hardware support to generate random numbers" on page 49.

---

6. (Optional step.) Create an **sshrc** file. If you need to run host-specific commands whenever a user logs in to this host, create an `/etc/ssh/sshrc` file. It is a shell script run only for SSH logins, not for non-SSH logins (such as rlogin or telnet). Examples of use are logging or running **ssh-agent**. If you do not need to do this, then do not create the file. If you create the file, it must be a shell script in `/bin/sh` syntax.

---

7. If the TCPIP.DATA file on the system is located in the UNIX file system, for example, named `/etc/resolv.conf`, copy `/etc/resolv.conf` to `/var/empty/etc/resolv.conf`.

   ```
   cp -p /etc/resolv.conf /var/empty/etc/resolv.conf
   ```

The OpenSSH daemon runs with privilege separation enabled by default. During privilege separation, the daemon cleaves itself into two processes, one with privileges and one without. The unprivileged user (the SSHD privilege separation user) handles network traffic and everything not requiring special privileges. This unprivileged process runs in a chroot jail of `/var/empty`. The **chroot** service changes the root directory from the current one to a new one; in this case, `/var/empty`. The root directory is the starting point for path searches of path names beginning with a slash. At some point, the privilege separation user invokes a TCP/IP system call which requires access to the TCPIP.DATA file. If this file is stored in the UNIX file system as `/etc/resolv.conf`, the privilege separation user will not have access to the file because it is not located off the new root file system of `/var/empty`. To make this file visible to the privilege separation user, the system administrator should copy `/etc/resolv.conf` to `/var/empty/etc/resolv.conf`.

Tip: Every time the installation changes the TCPIP.DATA statements, the TCPIP.DATA file must be recopied to the path name located off the `/var/empty` root, so that the updated information is found by the privilege separation user.

---

8. If your system is set up to run in another locale, see Chapter 7, "Globalization on z/OS systems," on page 55 for information about setting up your system or user environment.

When you are done, you have either created or edited the configuration files.

Setting up server authentication

The following are important notes for setting up server authentication.

1. To run **ssh-keyscan** against a host, the **sshd** daemon must be running on that host.
2. Verify all keys gathered via **ssh-keyscan** by displaying the key fingerprint with **ssh-keygen**.
3. For additional security, all host names and addresses can be hashed in the `ssh_known_hosts` file. The **ssh-keygen** and **ssh-keyscan** commands provide options for hashing host names and addresses.
4. If **ssh-keyscan** was not used to gather the host keys, then prepend the host name or address (for which the keys belong) to each key entry in the `ssh_known_hosts` file. **ssh-keyscan** automatically includes the host name or address in its output.
5. The system-wide `ssh_known_hosts` file is in the `/etc/ssh` directory.

Before the system administrator can start the **sshd** daemon, server authentication must be set up. During server authentication, when a client attempts to establish a secure connection with the server, keys are used to determine the trustworthiness of the server. Those keys can be stored in either UNIX files or SAF key rings, or both. For more information about storing the key rings, see "Choosing between UNIX files and key rings" on page 53.

You need to know whether you want to use SSH protocol version 1, protocol version 2, or both. Protocol version 2 is the default. Both protocols support similar authentication methods, but protocol version 2 is preferred because it provides additional mechanisms for confidentiality and integrity. Protocol version 1 lacks a strong mechanism for ensuring the integrity of the connection.

Restriction: If you are using SSH protocol version 1, you cannot use key rings to hold your keys. You must use UNIX files to hold RSA keys used for SSH protocol version 1.

The procedures for setting up server authentication are described in the following sections:

- "Steps for setting up server authentication when keys are stored in UNIX files"
- "Steps for setting up server authentication when keys are stored in key rings" on page 29

Steps for setting up server authentication when keys are stored in UNIX files

Perform the following steps to perform setup for server authentication if you are storing the keys in UNIX files.

1. Generate the host keys for the SSH server based on the protocol that you plan to use. (Host keys allow a client to verify the identity of the server.) The key files must be stored in the IBM-1047 (EBCDIC) code set. Assuming that the superuser running these commands is running in the default C locale, the key files are automatically stored in that code set.

   If you are using SSH protocol version 1, issue:

   ```
   ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_key -N ""
   ```

   If you are using SSH protocol version 2, issue:

   ```
   ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key -N ""
   ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ""
   ```

   The use of the **-N** option in the examples creates an empty passphrase for the host key. Host keys cannot have passphrases associated with them, because the daemon would have no way of knowing which passphrase to use with which host key.

   ---

2. Copy the local host's public keys to the `ssh_known_hosts` file at the remote host. The client uses the `ssh_known_hosts` file to verify the identity of the remote host.

   a. Log into the remote host.

   b. Append the local host's public keys to the `/etc/ssh/ssh_known_hosts` file at the remote host.

      If you are using SSH protocol version 1, use:

      ```
      /etc/ssh/ssh_host_key.pub
      ```

      If you are using SSH protocol version 2, use:

      ```
      /etc/ssh/ssh_host_dsa_key.pub
      /etc/ssh/ssh_host_rsa_key.pub
      ```

      You can use cut and paste to append the keys. Because a key is a long line, verify that the keys were not split across lines. Each key should be exactly one line of the file.

      If you use FTP to move your public key files to another system, treat the files as text to enable any necessary conversion between ASCII and EBCDIC.

   c. For each public key added to the remote `ssh_known_hosts` file, add the host name of the key to the start of the line. For more information, see "ssh_known_hosts file format" on page 122. All host names and addresses

in this file can be hashed for additional security. The **ssh-keygen** command provides the **-H** option for this purpose.

d. Log off the system. Clients logging into the host can now verify the identity of that host.

---

3. Gather the public host keys of remote hosts and store them in either a file or a certificate.

   a. If the remote hosts are not z/OS systems or if they are z/OS systems that do not use key ring support, use **ssh-keyscan** to redirect the resulting output to a file. Verify the keys in that file and add them to the previously created `/etc/ssh/ssh_known_hosts` file. If you do not verify the keys before creating the `/etc/ssh/ssh_known_hosts` file, users might be vulnerable to attacks. For additional security, the **ssh-keyscan** command provides the **-H** option to hash all host names and addresses in the output. See "ssh-keyscan" on page 111 for more information.

   b. If any remote hosts are z/OS systems with the host keys in a key ring, two methods of gathering and storing those keys on the local host are available. Either the public key is stored in the `/etc/ssh/ssh_known_hosts` file, or the public key is stored in a certificate associated with a key ring on the local host. That certificate is identified in the `/etc/ssh/ssh_known_hosts` file.

      1) Use **ssh-keyscan** as described earlier in this step, or

      2) Extract the public host keys from the remote host key ring as follows:

         - Use **ssh-keygen -e** on the remote host to export the public host key.

           Example:

           ```
           export _ZOS_SSH_KEY_RING_LABEL="SSHDAEM/SSHDring host-ssh-type"
           ssh-keygen -e > host-ssh-type.out
           ```

         - FTP the exported key to the local system.

         - Use **ssh-keygen -i** on the local system to import the public host key into a UNIX file.

           Example:

           ```
           ssh-keygen -i -f host-ssh-type.out >> /etc/ssh/ssh_known_hosts
           ```

When you are done, you have performed setup for server authentication in which keys will be stored in UNIX files. Each time the host keys are regenerated, they must be redistributed and added to the key ring of the remote system.

Figure 1 on page 29 shows how the known_hosts file is created when keys are stored in UNIX files.

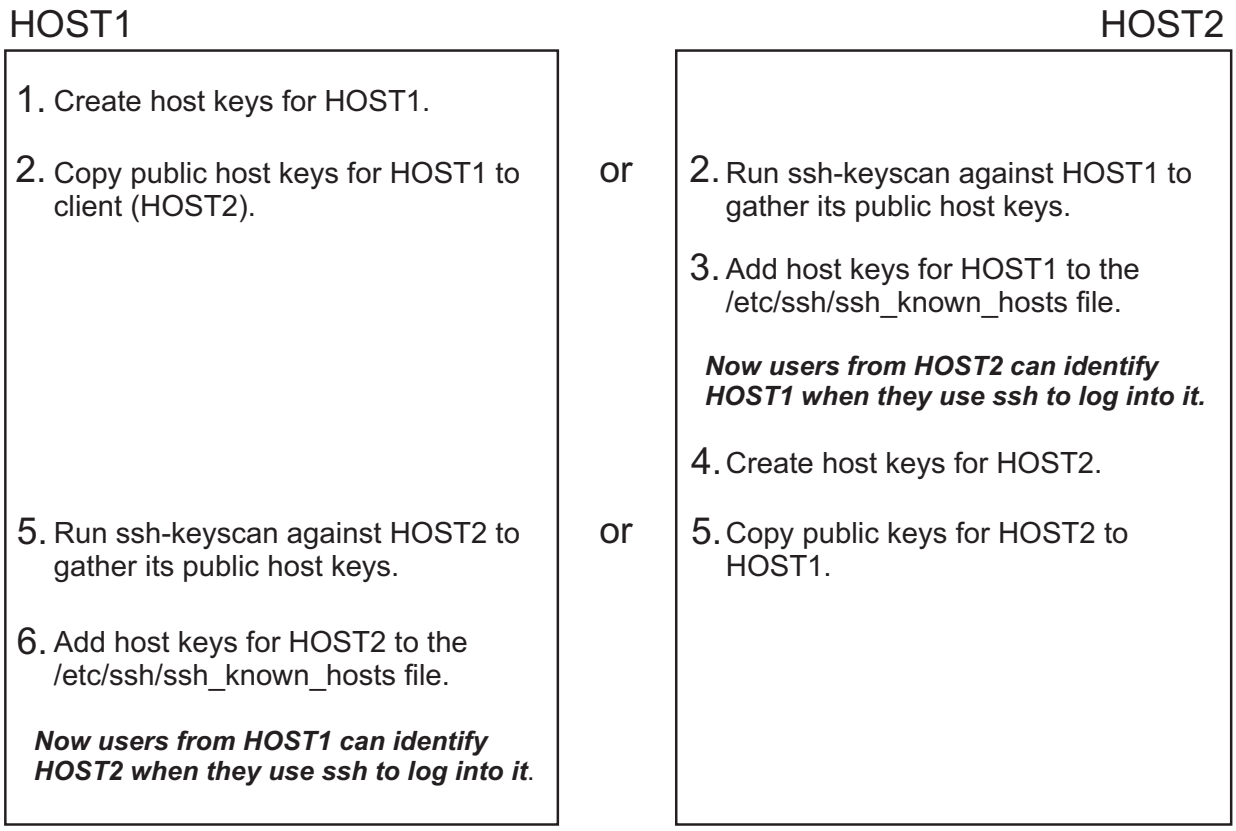

Figure 1. How the known_hosts file is created when keys are stored in UNIX files

Steps for setting up server authentication when keys are stored in key rings

The setup procedure has been divided into three steps:

- "Step 1: Generate the host keys for the SSH server" on page 30. Host keys allow a client to verify the identity of the server.
- "Step 2: Distribute the public keys from the local host to the remote hosts" on page 32. Clients use the `ssh_known_hosts` file to verify the identity of the remote host.
- "Step 3: Gather the public host keys of remote hosts" on page 34. Keys are verified and then added to the `/etc/ssh/ssh_known_hosts` file.

Use RACF® or a similar security product that supports key rings when storing key rings. SSH protocol version 2 is the only version that can be used when storing keys in key rings. If you want to use protocol version 1, then you must store the keys in UNIX files as described in "Steps for setting up server authentication when keys are stored in UNIX files" on page 27. Protocol version 2 provides additional mechanisms for confidentiality and integrity while protocol version 1 lacks a strong mechanism for ensuring the integrity of the connection. The key files must be stored in the IBM-1047 (EBCDIC) code set.

About the examples in this section

The examples provided for managing key rings and associated objects use the RACF RACDCERT command. If a different security product is used, consult that product's documentation to determine if it contains compatible support. For more information about the RACDCERT command, the necessary authority required to use the command, and any other options not described in this documentation, refer to *z/OS Security Server RACF Command Language Reference*.

In the examples, input names that are given in italics are variables that you can choose. Some of these names in italics contain hyphen characters (-) separating portions of the name. These hyphens are variable and are not required. The names given are merely suggestions and are consistently used throughout the examples. If you customize your own version in one step, that name will likely need to be used on other command steps as well.

The examples demonstrate using a self-signed certificate. Using a certificate chain, such as with root and intermediate certificate authority certificates, is supported. If you will be using more advanced certificate chains than the examples demonstrate, see "Validating certificates when using key rings" on page 54 for important considerations.

Step 1: Generate the host keys for the SSH server

Before you begin: You need to do the following:

- Make sure that a unique user ID that will be used to start the OpenSSH daemon has already have been set up on your system. A unique user ID is necessary because RACF uses the user ID, not the UNIX UID, for access control to key rings. The examples in this step use SSHDAEM as the user ID that starts the daemon and that also owns the associated host key rings. For more information about setting up the user ID that will be used to start the OpenSSH daemon, see "Starting sshd as a stand-alone daemon" on page 39.
- Determine whether you are working with real or virtual key rings because the setup steps vary depending on the type of key ring is being used. See *z/OS Security Server RACF Security Administrator's Guide* for more information about real and virtual key rings.

Perform the following steps to generate the host keys for the SSH server.

1. Create a real key ring if you do not yet have one to use for the host public keys. Omit this step if you plan to use a virtual key ring. Use the RACDCERT ADDRING command to create the new key ring, specifying the owning user ID and the key ring name. The ID keyword must specify the user ID that will be starting **sshd**. The key ring name can be any unique name for this user ID.

   Example:

   ```
   RACDCERT ID(SSHDAEM) ADDRING(SSHDring)
   ```

2. Using the RACDCERT GENCERT command, generate a host certificate with public and private keys based on the algorithms that are supported on the server (either RSA, DSA, or both). For RSA keys, the minimum size is 768 bits and the maximum size is 32768 bits. Typically, 2048 bits are considered

sufficient. DSA keys must be exactly 1024 bits as specified by FIPS 186-2. DSA keys larger than 1024 bits associated with certificates in a key ring are not supported by OpenSSH.

Do not use variant characters in the label name for the certificate. The **sshd** daemon must run only in the C locale and therefore interprets the key files (that is, the known host and authorized key files) as encoded in code set IBM-1047.

Examples: Although the following examples demonstrate how to create non-ICSF (Integrated Cryptographic Storage Facility) certificates in the RACF database, ICSF can also be used to store the certificate and associated keys for RSA only. These keys can be generated by software using ICSF or by hardware using PCI Cryptographic Coprocessor (PCICC). For more information, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

- To generate a certificate and an RSA public/private key pair, storing the private key in the RACF database as a non-ICSF key:

  ```
  RACDCERT GENCERT ID(SSHDAEM) SUBJECTSDN(CN('host-ssh-rsa-cn'))
  SIZE(2048) WITHLABEL('host-ssh-rsa')
  ```

- To generate a certificate and a DSA public/private key pair, storing the private key in the RACF database as a non-ICSF key:

  ```
  RACDCERT GENCERT ID(SSHDAEM) SUBJECTSDN(CN('host-ssh-dsa-cn'))
  SIZE(1024) DSA WITHLABEL('host-ssh-dsa')
  ```

The SUBJECTSDN parameter offers several more customizable keywords, which are not shown in the preceding examples, that can be included in the distinguished name. The label assigned to the certificate must be chosen to be unique within the RACF database. The user ID must match the owner of the key ring.

---

3. If real key rings are being used, use the RACDCERT CONNECT command to connect the certificate to the host key ring. Omit this step if you plan to use virtual key rings. You must identify the user ID that owns the certificate and the user ID that owns the key ring. These are typically the same for this connect command.

   Example:

   ```
   RACDCERT CONNECT(ID(SSHDAEM) LABEL('host-ssh-type')
    RING(SSHDring) USAGE(PERSONAL)) ID(SSHDAEM)
   ```

---

4. Add a line in the z/OS-specific `zos_sshd_config` file for each certificate being used for a host key.

   - **For real key rings,** add the following line:

     ```
     HostKeyRingLabel "SSHDAEM/SSHDring host-ssh-type"
     ```

   - **For virtual key rings,** add the following line:

     ```
     HostKeyRingLabel "SSHDAEM/* host-ssh-type"
     ```

---

5. Restrict access to the key ring. To prevent access to the host private keys by any other user, permit only the user ID (for example, SSHDAEM) that starts the **sshd** daemon. See "Managing key rings and restricting access to them" on page 53 for more information.

   Examples:

   - To prohibit universal access to SSHDring, using ring-specific profile checking:

```
RDEFINE RDATALIB SSHDAEM.SSHDring.LST UACC(NONE)
PERMIT SSHDAEM.SSHDring.LST CLASS(RDATALIB) ID(SSHDAEM) ACCESS(READ)
```

If the RDATALIB class is not yet active and RACLISTed:

```
SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
```

Refresh the class:

```
SETROPTS RACLIST(RDATALIB) REFRESH
```

- To prohibit universal access to the SSHDAEM user's virtual key ring, using ring-specific profile checking:

```
RDATALIB SSHDAEM.IRR_VIRTUAL_KEYRING_LST UACC(NONE)
PERMIT SSHDAEM.IRR_VIRTUAL_LISTRING_LST CLASS(RDATALIB) ID(SSHDAEM) ACCESS(READ)
```

If the RDATALIB class is not yet active and RACLISTed:

```
SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
```

Refresh the class:

```
SETROPTS RACLIST(RDATALIB) REFRESH
```

- To prohibit universal access to any key ring on the system, using global profile checking:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
```

If the FACILITY class is not yet active and RACLISTed:

```
SETROPTS RACLIST(FACILITY) CLASSACT(FACILITY)
```

Refresh the class:

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

When you are done with Step 1, you have generated the host keys for the SSH server. Now go to "Step 2: Distribute the public keys from the local host to the remote hosts."

Step 2: Distribute the public keys from the local host to the remote hosts

Step 2 is intended for remote hosts that use key rings. If a remote host does not use key rings, then use **ssh-keygen** to distribute the public host keys as described in Step 3 in "Steps for setting up server authentication when keys are stored in UNIX files" on page 27.

Perform the following steps to distribute the public keys from the local host to the `ssh_known_hosts` file on the remote host.

1. Export each certificate in DER format without the private key into a data set using the RACDCERT EXPORT command. Specify the certificate identification and request CERTDER for the export format. Choose a data set to store the exported certificate and specify it on the DSN parameter. If the data set specified for DSN already exists, it is deleted and reallocated by the RACDCERT EXPORT command.

   Example:

   ```
   RACDCERT EXPORT(LABEL('host-ssh-type')) ID(SSHDAEM)
     FORMAT(CERTDER) DSN('host.sshcert.type')
   ```

---

2. Use FTP to distribute each exported certificate data set in binary format to the remote hosts.

---

3. On the remote host, if real key rings are being used, create a new key ring if you do not yet have a key ring to use for the known host public keys. Omit this step if you plan to use virtual key rings. Use the RACDCERT ADDRING command, specifying the owning user ID and the key ring name. If you have not yet created the user ID that will be starting the **sshd** daemon on this remote host, do that first. The user ID specified here must be the user ID that will be running the **sshd** daemon on this remote host which is assumed to be SSHDAEM in the following examples. The key ring name can be any unique name for this user ID.

   Example:

   ```
   RACDCERT ID(SSHDAEM) ADDRING(SSHKnownHostsRing)
   ```

---

4. On the remote host, use the RACDCERT ADD command to add the exported certificate on the remote host. Specify the data set that you distributed to this remote host by using FTP. Also specify the user ID that should own the certificate and indicate that this certificate is trusted. The user ID specified here must be the user ID that will be running the **sshd** daemon on this remote host which is assumed to be SSHDAEM in the following examples. You will also specify the label for this certificate on this remote host. This label must be unique for the user ID within the RACF database and is used to identify this certificate on future commands and for reference as a known host certificate.

   This certificate contains only the public key.

   Example:

   ```
   RACDCERT ADD('host.sshcert.type') ID(SSHDAEM)
     WITHLABEL('host-ssh-type') TRUST
   ```

---

5. On the remote host, if real key rings are being used, use the RACDCERT CONNECT command to connect each certificate into the known hosts key ring. Omit this step if you plan to use virtual key rings. You must identify the user ID that owns the certificate and the user ID that owns the key ring. These will typically be the same for this connect command.

   Example:

   ```
   RACDCERT CONNECT(ID(SSHDAEM) LABEL('host-ssh-type')
      RING(SSHKnownHostsRing)) ID(SSHDAEM)
   ```

---

6. On the remote host, edit the system-wide known_hosts file `/etc/ssh/ssh_known_hosts` to add a line for each host certificate connected in Step 4. The line must contain the host name or host names followed by *zos-key-ring-label="KeyRingOwner/KeyRingName label*".

   Examples:

   - **For a real key ring** (for example, SSHKnownHostsRing), add:

     ```
     host zos-key-ring-label="SSHDAEM/SSHKnownHostsRing host-ssh-type"
     ```

   - **For a virtual key ring** (for example, one owned by SSHDAEM), add:

     ```
     host zos-key-ring-label="SSHDAEM/* host-ssh-type"
     ```

   For more information, see the **sshd** command section “ssh_known_hosts file format” on page 122.

7. On the remote host, permit user access to the known hosts key ring. All OpenSSH client users on this system must have authority to read the public keys from this key ring. For details about the methods of permitting access, see "Managing key rings and restricting access to them" on page 53.

   Examples:

   - To define universal access to the real key ring, SSHKnownHostsRing, using ring-specific profile checking:

     ```
     RDEFINE RDATALIB SSHDAEM.SSHKnownHostsRing.LST UACC(READ)
     ```

     If the RDATALIB class is not yet active and RACLISTed:

     ```
     SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
     ```

     Refresh the class:

     ```
     SETROPTS RACLIST(RDATALIB) REFRESH
     ```

   - To define universal access to the SSHDAEM user's virtual key ring, using ring-specific profile checking:

     ```
     RDEFINE RDATALIB SSHDAEM.IRR_VIRTUAL_KEYRING_LST UACC(READ)
     ```

     If the RDATALIB class is not yet active and RACLISTed:

     ```
     SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
     ```

     Refresh the class:

     ```
     SETROPTS RACLIST(RDATALIB) REFRESH
     ```

   - To define (and permit) universal access to any key ring on the system, using global profile checking:

     ```
     RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(UPDATE)
     ```

     If the FACILITY class is not yet active and RACLISTed:

     ```
     SETROPTS RACLIST(FACILITY) CLASSACT(FACILITY)
     ```

     Refresh the class:

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

8. Log off the remote host.

When you are done with Step 2, you have distributed the public keys on the local host to the remote hosts. Now go to "Step 3: Gather the public host keys of remote hosts."

Step 3: Gather the public host keys of remote hosts

Step 3 is intended for remote hosts that use key rings. If a remote host does not use key rings, then use **ssh-keyscan** to gather the public host keys, as described in Step 3 on page 28 in Steps for setting up server authentication when keys are stored in UNIX files.

1. Create a new key ring if you do not yet have one to use for the host public keys on your local host. Omit this step if you plan to use virtual key rings. Use the RACDCERT ADDRING command, specifying the owning user ID and

the key ring name. The ID keyword should specify the user ID that will be starting **sshd**. The key ring name can be any unique name for the specified user ID.

Example:

```
RACDCERT ID(SSHDAEM) ADDRING(SSHKnownHostsRing)
```

---

2. On the remote host, export each host key certificate in DER format without the private key and use FTP to distribute it in binary format to the local host. The RACDCERT EXPORT command can perform this type of export. Specify the certificate identification and request CERTDER for the export format. Choose a data set to store the exported certificate and specify it on the DSN parameter. If the data set specified for DSN already exists, it is deleted and reallocated by the RACDCERT EXPORT command.

   Example:

   ```
   RACDCERT EXPORT(LABEL('host-ssh-type')) ID(SSHDAEM)
     FORMAT(CERTDER) DSN('host.sshcert.type')
   ```

---

3. Use FTP to distribute each data set in binary format from the remote host to the local host.

---

4. On the local host, add each certificate into the SAF database. Use the RACDCERT ADD command to add the exported certificate on the remote host. Specify the data set that you copied from the local host using FTP, the user ID that should own the certificate, and indicate that this certificate is trusted. The user ID specified here must be the user ID that will be running the **sshd** daemon on this local host. You will also be specifying the label for this certificate on this local host. This label must be unique for the user ID within the RACF database, and will be used to identify this certificate on future commands and for reference as a known host certificate.

   This certificate will contain only the public key.

   Example:

   ```
   RACDCERT ADD('host.sshcert.type') ID(SSHDAEM)
     WITHLABEL('host-ssh-type') TRUST
   ```

---

5. Connect each certificate into the known hosts key ring if a real key ring is being used. Omit this step if you plan to use virtual key rings. The RACDCERT CONNECT command can be used. You must identify the user ID that owns the certificate and the user ID that owns the key ring. These will typically be the same for this connect command.

   Example:

   ```
   RACDCERT CONNECT(ID(SSHDAEM) LABEL('host-ssh-type')
     RING(SSHKnownHostsRing)) ID(SSHDAEM)
   ```

---

6. Edit the local host's system-wide known_hosts file `/etc/ssh/ssh_known_hosts` to add a line for each of the host certificates imported in Step 4. The line must contain the host name or host names followed by *zos-key-ring-label="KeyRingOwner/KeyRingName label*".

   Example:

   - **If a real key ring is being used** (for example, SSHKnownHostsRing), issue:

     ```
     mvshost zos-key-ring-label="SSHDAEM/SSHKnownHostsRing host-ssh-type"
     ```

- **If a virtual key ring is being used** (for example, one owned by SSHDAEM), issue:

  ```
  mvshost zos-key-ring-label="SSHDAEM/* host-ssh-type"
  ```

  For more information, see the **sshd** command section "ssh_known_hosts file format" on page 122.

---

7. On the local host, permit user access to the known hosts key ring. For details about the methods of permitting access, see Step 7 on page 34 in "Step 2: Distribute the public keys from the local host to the remote hosts" on page 32.

When you are done with Step 3, you have gathered the public host keys of remote hosts and edited the local `/etc/ssh/ssh_known_hosts` file to include the imported host certificates. Now clients can verify the identity of remote hosts. Each time the host keys are regenerated in the key ring, they must be redistributed and added to the key ring of the remote system.

Figure 2 on page 37 shows a high-level view of the operations performed to set up the server's host keys when they are stored in real key rings.

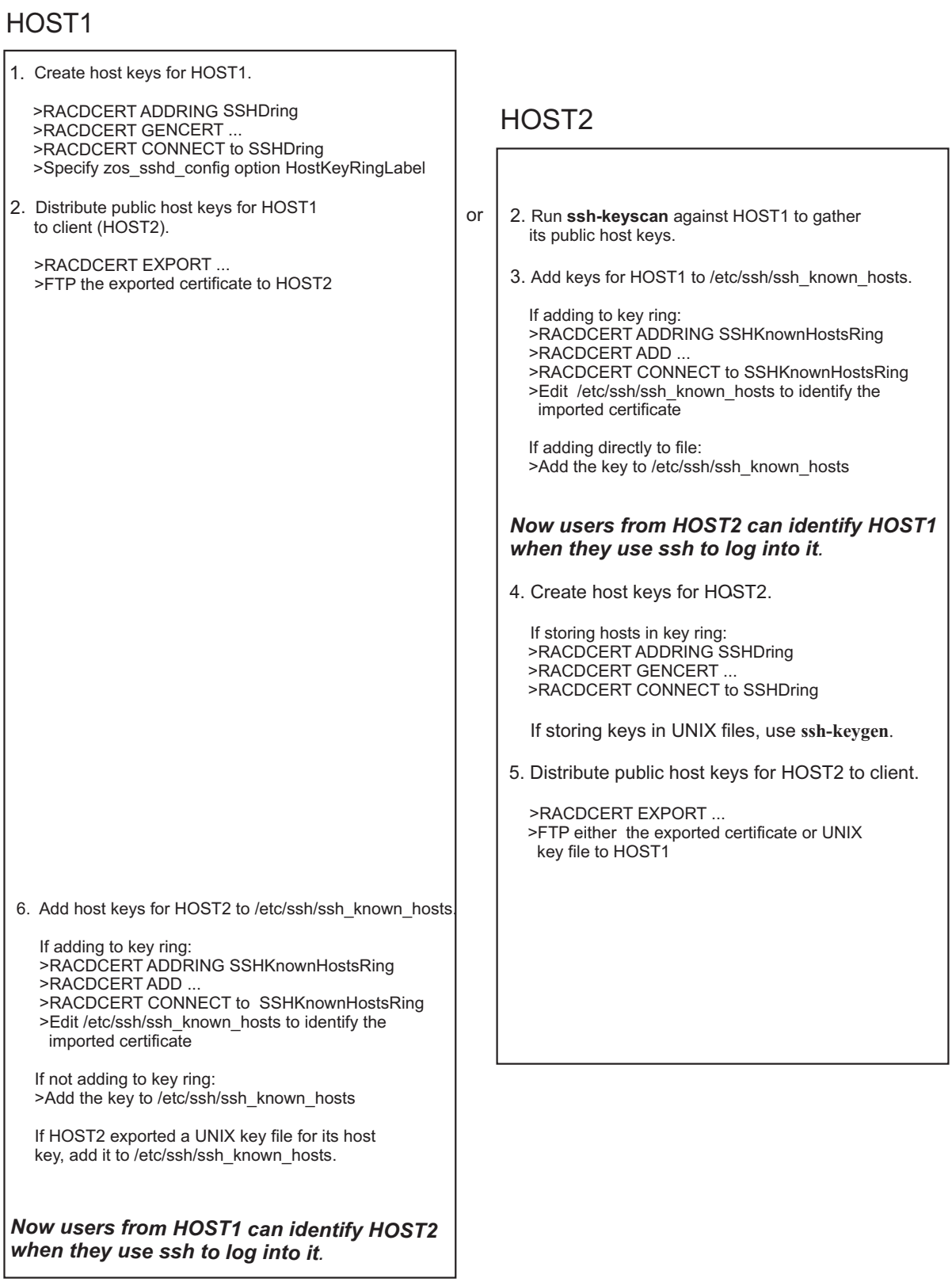

Figure 2. How the server's host keys are set up when they are stored in real key rings

Step for creating the sshd privilege separation user

Privilege separation (where the OpenSSH daemon creates an unprivileged child process to handle incoming network traffic) is enabled in the default configuration for **sshd**.

Before you begin: You need to know the new group ID and unused nonzero user ID that you want to use. The user ID and group ID for the privilege separation user "SSHD" is not the same user ID that will be used to start the OpenSSH daemon. The user ID you choose for the SSHD user should be unprivileged.

You must also be logged onto TSO/E with RACF SPECIAL authority. (Instead of using RACF, you could use an equivalent security product if it supports the SAF interfaces required by z/OS UNIX, which are documented in *z/OS Security Server RACF Callable Services*.)

Perform the following step to create the **sshd** privilege separation user.

- Set up a user account for the **sshd** privilege separation user by issuing the following commands where *xxx* is an unused group ID, and *yyy* is an unused nonzero user ID.

  ```
  ADDGROUP SSHDG OMVS(GID(xxx))
  ADDUSER SSHD DFLTGRP(SSHDG) OMVS(UID(yyy) HOME('/var/empty')
    PROGRAM('/bin/false')) NOPASSWORD
  ```

  Tip: If you have a user ID naming policy that does not allow you to assign this user as "SSHD", you can create an "sshd" entry in the user ID alias table, and map it to the user ID that was actually defined. See *z/OS UNIX System Services Planning* for more information about the user ID alias table.

When you are done, you have created the **sshd** privilege separation user.

Setting up the message catalog for IBM Ported Tools for z/OS: OpenSSH

Setting up the message catalog for IBM Ported Tools for z/OS: OpenSSH is an optional task. To see message numbers (for example, FOTSnnnn) associated with OpenSSH error messages, no special OpenSSH message catalog setup is required. If you do not want to see message numbers, then you must set the environment variable _ZOS_OPENSSH_MSGCAT="NONE" before running an OpenSSH command. This setting can be applied to all shell users by exporting it from the default system-wide user environment files, `/etc/profile` and `/etc/csh.cshrc`. The _ZOS_OPENSSH_MSGCAT environment variable identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

Table 13. Values for the _ZOS_OPENSSH_MSGCAT environment variable

Value	Result
"openssh.cat"	Message numbers are associated with OpenSSH error messages by default.
"openssh"	Message numbers are associated with OpenSSH error messages if the NLSPATH environment variable includes the following path: /usr/lib/nls/msg/%L/%N.cat.
"NONE"	Message numbers are not associated with OpenSSH error messages.
Unset or set to an invalid value	Message numbers are associated with OpenSSH error messages by default.

Starting the sshd daemon

You can start the **sshd** daemon in one of two ways:

- As a stand-alone daemon, as described in "Starting sshd as a stand-alone daemon." As a stand-alone daemon, **sshd** listens for TCP connections on a port (default 22), and starts child processes to handle the requested connections.
- As a daemon running under **inetd**, as described in "Starting sshd as a daemon running under inetd" on page 42. The **inetd** program listens on the specified port and starts an instance of the **sshd** daemon for each requested connection.

Starting sshd as a stand-alone daemon

The **sshd** daemon can be started as a stand-alone daemon.

This setup assumes that RACF is used as your security product. If you use a different security product, you need to determine the equivalent setup for that product. You also need RACF SPECIAL (administrator) authority to perform the RACF setup.

You need to decide which user ID will be used to start the daemon. The user ID might already have been set up on your system.

Rules:

- The user ID must have a UID of 0 and ACCESS(READ) permission to BPX.DAEMON.
- Do not choose "SSHD" as the user name to assign to the daemon. The user name "SSHD" is reserved for the privilege separation user, which is not a UID(0) user ID.
- If the host system has the BPX.POE resource in the FACILITY class defined, the UID invoking the OpenSSH daemon must have ACCESS(READ) permission.
- If **ssh-rand-helper** is used to generate random numbers, the user ID must have write access to its home directory in order to store temporary seed files generated by **ssh-rand-helper.** Refer to "Using hardware support to generate random numbers" on page 49 for more information about random number generation.
- If the SERVAUTH class is active, the user ID might need to be authorized to some of the network resources protected by the SERVAUTH class. For more information about the SERVAUTH class, see *z/OS Communications Server: IP Configuration Guide*.

Example: The following example assumes that the SSHDAEM user ID is defined as UID(0) and has READ access to the BPX.DAEMON profile in the FACILITY class. It also assumes that the SSHDAEM user ID was set up like the OMVSKERN user ID. For more information about how to set up OMVSKERN, see the section on preparing RACF in *z/OS UNIX System Services Planning*.

```
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST(FACILITY)
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(SSHDAEM) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

The section on establishing the correct level of security for daemons in *z/OS UNIX System Services Planning* discusses the z/OS UNIX level of security.

Ways to start sshd as a stand-alone daemon

There are several ways to start and restart **sshd**. The method used depends on the level of control that the installation has chosen for daemons.

Using BPXBATCH

You can start **sshd** with a cataloged procedure by using BPXBATCH to invoke a daemon program located in the z/OS UNIX file system. If you use BPXBATCH as a started procedure to initiate the SSHD job, it will complete typically with a return code of CC=0. A forked copy of the daemon will be left running, which is normal.

These steps explain what to do.

1. Create a cataloged procedure.

   Example: Following is a sample procedure:

   ```
   //SSHD    PROC
   //SSHD    EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
   //        PARM='PGM /bin/sh -c /etc/ssh/sshd.sh'
   //* STDIN and STDOUT are both defaulted to /dev/null
   //STDERR  DD PATH='/tmp/sshd.stderr',
   //           PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU)
   ```

   The following is the sample shell script to be used with the preceding sample procedure. The sample procedure assumes that this sample shell script is stored in `/etc/ssh/sshd.sh` and is executable by the caller (for example, `chmod 700 /etc/ssh/sshd.sh`).

   ```
   #!/bin/sh
   export _EDC_ADD_ERRNO2=1
   nohup /usr/sbin/sshd -f /etc/ssh/sshd_config &
   sleep 1
   ```

   Specifying REGION=0M in the JCL is equivalent to specifying MEMLIMIT=NOLIMIT. Options for altering this behavior include utilizing IEFUSI to set MEMLIMIT ceilings for your system because IEFUSI settings override the JCL. Alternatively, you can use SMFPRMxx system default settings, but this works only if there are no REGION or MEMLIMIT specifications in the JCL.

2. For this **sshd** cataloged procedure to obtain control with superuser and daemon authority, you must add it to the STARTED class.

   The procedure in this example is named "SSHD" because it starts the **sshd** daemon. It should not be confused with the SSHD privilege separation user, which is an unprivileged user ID that the daemon uses to execute unprivileged areas of code.

   Example: This example assumes that the SSHDAEM user ID is defined as UID(0), and has READ access to the BPX.DAEMON profile in the FACILITY class. For more information about how to set up SSHDAEM, see "Starting sshd as a stand-alone daemon" on page 39. Following is an example of a cataloged procedure:

   ```
   SETROPTS GENERIC(STARTED)
   RDEFINE STARTED SSHD.* STDATA(USER(SSHDAEM)
   GROUP(OMVSGRP) TRUSTED(NO)
   SETROPTS RACLIST(STARTED) REFRESH
   ```

   The section about using started procedures in *z/OS Security Server RACF Security Administrator's Guide* contains more information about using started procedures and the STARTED class.

3. To start **sshd**, issue the following command from the MVS console:

   ```
   S SSHD
   ```

You should see the message IEF695I on the MVS syslog. The user ID indicated in the message should be defined as UID(0) with READ access to the BPX.DAEMON profile in the FACILITY class. The group indicated in the message should have an OMVS segment containing a GID value. With the default values from Step 2 on page 40 (SSHDAEM and OMVSGRP), the message would look like this:

```
IEF695I START SSHD     WITH JOBNAME SSHD   IS ASSIGNED TO
USER SSHDAEM   ,GROUP OMVSGRP
```

The user ID and group must not be SSHD and SSHDG because this would indicate that the daemon was started with the SSHD privilege separation user.

Whenever the **sshd** daemon is terminated, you can issue `S SSHD` to restart it.

Using the /etc/rc shell script

You can put the command in the `/etc/rc` shell script to start the daemon automatically during initialization. For information about starting programs from `/etc/rc`, see the section on customizing `/etc/rc` in *z/OS UNIX System Services Planning*.

When UNIX systems are initialized (IPLed or restarted), the `/etc/rc` shell script is run to perform system initialization functions and to start daemons. If a daemon terminates, a superuser must restart the daemon.

To start **sshd** from the `/etc/rc` shell script, add the following to the `/etc/rc` file:

```
_BPX_JOBNAME=SSHD /usr/sbin/sshd &
```

In this example, the _BPX_JOBNAME environment variable is set to assign a job name of SSHD to the **sshd** daemon. Doing so allows the operator to have better control over managing the **sshd** daemon.

When started from the`/etc/rc` shell script, stdin and stdout are set to **/dev/null** and stderr is set to **/etc/log** for recording any errors. If you want to separate the standard error of **sshd** from that of all `/etc/rc` error output, you can specify the **sshd** command to redirect standard error as follows:

```
_BPX_JOBNAME=SSHD /usr/sbin/sshd 2>/tmp/sshd.stderr &
```

If the **sshd** daemon process is stopped, it must be started by a user ID with appropriate privileges. For more information about setting up the user ID that will be used to start the OpenSSH daemon, see "Starting sshd as a stand-alone daemon" on page 39.

From the shell

If you are running with UNIX-level security, (for example, without BPX.DAEMON), you can start **sshd** from a superuser ID in the UNIX shell. This security level is not generally adequate for z/OS systems.

Issue:

```
_BPX_JOBNAME=SSHD /usr/sbin/sshd &
```

For an explanation about using &, see *z/OS UNIX System Services Planning*.

Restarting the sshd daemon without bringing it down

If the server configuration files are changed after the **sshd** daemon is running, the changes do not affect the daemon, unless a SIGHUP signal is sent to the daemon

process. To restart the **sshd** daemon, reading the configuration files, including z/OS-specific files, without terminating existing SSH connections, issue

```
kill -s HUP $(cat /var/run/sshd.pid)
```

The name of the `/var/run/sshd.pid` file can be changed by using the **sshd_config** keyword PidFile.

SIGHUP does not reset command-line options (which might override the configuration files). If you want to change a command-line option, the daemon will have to be stopped and then restarted with the new command-line option.

Starting sshd as a daemon running under inetd

You can start the **sshd** daemon as a daemon running under **inetd**.

Steps for starting the sshd daemon under inetd

Before you begin: You need to be familiar with **inetd** configuration. You should also be aware that starting **sshd** through **inetd** could decrease performance of **ssh** connection startup time on your system. For every **ssh** connection started, **inetd** will start a new **sshd**. The **sshd** daemon startup incurs some overhead due to basic initialization and protocol version 1 server key generation.

Perform the following steps to start the **sshd** daemon under **inetd**.

1. In the TCP/IP services configuration file, add an entry to establish the connection between TCP/IP and z/OS UNIX. This is the `/etc/services` file or the hlq.ETC.SERVICES data set, where **hlq** is the prefix defined by DATASETPREFIX in the TCP/IP profile "TCPIP" by default). The format is:

   ```
   ssh 22/tcp
   ```

   ---

2. In the `/etc/inetd.conf` file, add a line similar to the following:

   ```
   ssh stream   tcp      nowait   SSHDAEM  /usr/sbin/sshd  sshd -i
   ```

   The **-i** option specifies **inetd** behavior, with a single connection on a TCPIP socket attached to **sshd**'s stdin and stdout.

When you are done, you have started the **sshd** daemon under **inetd**.

Restarting the sshd daemon under inetd without bringing it down

If **inetd** is currently running, send it a SIGHUP signal to allow the new configuration files with **sshd** settings to be read.

Stopping the sshd daemon

To stop the **sshd** daemon from the MVS console, follow these steps:

1. Determine the address space ID (ASID) of the **sshd** process. Issue:

   ```
   D A,SSHD*
   ```

   The ASID of the SSHD daemon will be returned.
2. Using the ASID obtained in Step 1, determine the process ID (PID) of the **sshd** process. Issue:

   ```
   D OMVS,ASID=aaaa
   ```

where *aaaa* is the ASID obtained in Step 1 on page 42. The PID of the daemon will be returned.

3. Using the PID obtained in Step 2 on page 42, stop the **sshd** daemon. Issue:

   ```
   F BPXOINIT,TERM=pppppppp
   ```

   where *pppppppp* is the PID obtained in Step 2 on page 42.

To stop **sshd** from z/OS UNIX, follow these steps:

1. Determine the process ID (PID) of the **sshd** daemon by looking at the contents of the file `/var/run/sshd.pid`. By default, the **sshd** PID is written to `/var/run/sshd.pid` when **sshd** is started. The name of the `/var/run/sshd.pid` file can be changed by using the **sshd_config** keyword PidFile. To find the PID, issue:

   ```
   cat /var/run/sshd.pid
   ```

   The PID of the **sshd** daemon will be returned.

2. Issue the z/OS UNIX **kill** command against the PID that was obtained in Step 1. For example:

   ```
   kill $(cat /var/run/sshd.pid)
   ```

   or

   ```
   kill pppppppp
   ```

   where *pppppppp* is the PID obtained in Step 1.

To stop the **sshd** daemon with a cataloged procedure using BPXBATCH, follow these steps:

1. Create a cataloged procedure. For example:

   ```
   //STOPSSHD PROC
   //STOPSSHD EXEC PGM=BPXBATCH,
   // PARM='PGM /bin/sh -c /etc/ssh/stopsshd.sh'
   //* STDIN and STDOUT are both defaulted to /dev/null
   //STDERR DD PATH='/tmp/sshd.stderr',
   // PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU)
   ```

   The following is the sample shell script to be used with the preceding sample procedure. The sample procedure assumes that this sample shell script is stored in the `/etc/ssh/stopsshd.sh` file and is executable by the caller (for example, `chmod 700 /etc/ssh/stopsshd.sh`).

   ```
   #!/bin/sh
   kill $(cat /var/run/sshd.pid)
   ```

   By default, the **sshd** PID is written to the `/var/run/sshd.pid` file when **sshd** is started. If the name of the **sshd** PID file was changed by using the **sshd_config** PidFile keyword then this sample shell script must be changed accordingly. (The keyword is described in "PidFile" on page 155.)

2. For the cataloged procedure to obtain control with superuser and daemon authority, you must add it to the STARTED class.

   Example: This example assumes that the SSHDAEM user ID is defined as UID(0) and has READ access to the BPX.DAEMON profile in the FACILITY class. For more information about how to set up SSHDAEM, see "Starting sshd as a stand-alone daemon" on page 39.

   ```
   SETROPTS GENERIC(STARTED)
   RDEFINE STARTED STOPSSHD.* STDATA(USER(SSHDAEM)
   GROUP(OMVSGRP) TRUSTED(NO))
   SETROPTS RACLIST(STARTED) REFRESH
   ```

The section about using started procedures in *z/OS Security Server RACF Security Administrator's Guide* contains more information about using started procedures and the STARTED class.

3. To stop the **sshd** daemon, issue the following command from the MVS console:

   ```
   S STOPSSHD
   ```

   Whenever the sshd daemon is started, you can issue S STOPSSHD to stop it.

Running the sshd daemon in a multilevel-secure environment

The OpenSSH daemon (**sshd**) can be used on a multilevel-secure system to control a user's security label at login. Review *z/OS Planning for Multilevel Security and the Common Criteria* before using the daemon on a multilevel-secure system.

The OpenSSH daemon will attempt to derive a security label from the user's port of entry, as defined in a NetAccess profile. To successfully login to a multilevel-secure system, the login user ID must be permitted to the security label defined in the NetAccess profile for the client IP address. These checks are performed for any user invoking **ssh**, **scp**, or **sftp** to perform remote operations on the multilevel-secure system. For more information about NetAccess profiles and running daemons in a multilevel-secure environment, see *z/OS Communications Server: IP Configuration Guide*.

Verifying security labels for directories

Verify that the following directories have been assigned the appropriate security labels.

Directory	Permission	Owner	Security label
/var/empty	755	UID(0)	SYSHIGH
/var/run	755	UID(0)	SYSLOW
/usr/lib/ssh	755	UID(0)	SYSLOW
/etc/ssh	755	UID(0)	SYSLOW

Configuring sshd for multilevel security

The OpenSSH daemon must be started by a UID(0) user ID running with a security label of SYSMULTI, and the user ID must be authorized to the SERVAUTH NETACCESS profiles. The privilege separation user ("SSHD") must be assigned and permitted to the SYSMULTI seclabel. Assign a security label of SYSHIGH to the `/var/empty` directory.

If the host system has the BPX.POE resource in the FACILITY class defined, the UID invoking the OpenSSH daemon must have ACCESS(READ) permission.

Guidelines: In a multilevel-secure environment:

1. **sshd** should not be invoked through **inetd**.
2. Port forwarding should be disabled because it could allow a user to bypass NetAccess profile settings. It is disabled by default. See the description of the **sshd_config** keywords "AllowTcpForwarding" on page 145 and "X11Forwarding" on page 157.

If users are attempting login with password authentication and do not have authorization to log in from their IP address, then the login will fail at password

entry and a message should be written to the MVS console by the security product. If they are attempting login via public key authentication and do not have authorization to log in from their IP address, the attempted login will be terminated before the users enter a passphrase.

The following output is a sample failure of a client public key authentication in a multilevel-secure environment:

```
debug3: send_pubkey_test
debug2: we sent a publickey packet, wait for reply
Connection closed by UNKNOWN
```

The OpenSSH daemon writes an error message to the UNIX syslog for these failures.

Considerations for running the OpenSSH daemon when TERMINAL classes are defined

The OpenSSH daemon recognizes TERMINAL class settings.

- If the user is attempting login with password authentication and does not have authorization to log in from their terminal, then the login will fail at password entry and a message should be written to the MVS console by the security product.
- If the user is attempting login via public key authentication and does not have authorization to log in from their terminal, the attempted login will be terminated before the user enters a passphrase.

The following output is a sample client public key authentication failure when a TERMINAL class is enabled:

```
debug3: send_pubkey_test
debug2: we sent a publickey packet, wait for reply
Connection closed by UNKNOWN
```

The OpenSSH daemon writes an error message to the UNIX syslog for these failures.

Limiting file system name space for sftp users

Some administrators might want to limit the file system name space that is accessible by users during file transfer operations. This task can be accomplished by configuring the **sshd** daemon to change the root directory of the **sftp** user connection. The administrator uses the **sshd_config** keyword ChrootDirectory to set up the environment. The keyword is described in “ChrootDirectory” on page 146.

After the environment has been set up, searches for file system objects (files and directories) are relative to the user's new root directory. If the new root directory does not contain a duplicate of the required programs or support files needed by the user, then the session might not be usable. The "internal-sftp" subsystem can be used to overcome this setup problem for **sftp** users. Specifying "internal-sftp" on either the **sshd_config** keywords Subsystem or ForceCommand causes the **sshd** daemon to implement an in-process **sftp** server. Such a server does not require duplication of the **sftp-server** command or other support files in the new root directory in order to connect via **sftp**. Thus, combining the use of the **sshd_config**

keyword ChrootDirectory and the "internal-sftp" subsystem enables full **sftp** file transfer functionality, while limiting the file system objects that are accessible to the user. (The two keywords are described in “Subsystem” on page 156 and “ForceCommand” on page 149.)

Note that specifying "internal-sftp" on the **sshd_config** keyword ForceCommand enables an in-process **sftp** server to be the only command to be run, regardless of the command specified by the user. For example, this prevents the user from running **scp** or from starting an interactive shell session via **ssh** on the server. In addition, the in-process **sftp** server allows users without shell access on the server to still transfer files via **sftp**. Using the ForceCommand keyword in this manner allows the administrator to apply this restriction to a limited set of users when placed inside a Match keyword as described in “Match” on page 152.

Public key authentication can also be used with the **sshd_config** keyword ChrootDirectory. However, the **sshd** daemon will search for the user's public keys (see the **sshd_config** keyword “AuthorizedKeysFile” on page 146) starting from the original root directory, not the new root directory specified by the ChrootDirectory keyword. Therefore, depending on the location of the new root directory, the user might not have access to their own public keys used during authentication.

Example 1: Use the **sshd_config** keyword ChrootDirectory and "internal-sftp" subsystem to cause the **sshd** daemon to set a user's root directory to the user's home directory.

Server (name is "server1") **sshd_config** keywords:

```
Subsystem sftp internal-sftp
   ChrootDirectory %h
```

Client (user "employee1", home directory is `/u/employee1`):

```
> sftp server1
 Connecting to server1...
 sftp> pwd
 Remote working directory: /
 sftp> ls -a
   .        ..              .profile     .sh_history
 .ssh         myfile
```

After connecting and setting the root directory, the **sshd** daemon also attempts to change the user's current working directory to the user's home directory, relative to the root directory that is now in effect. For example, if the user's home directory were `/u/employee1`, then the **sshd** daemon would attempt to set the user's current working directory relative to the root directory (which also happens to be `/u/employee1`). Therefore, the **sshd** daemon sets the user's current working directory to `/u/employee1/u/employee1`, if the directory exists. This action might or might not be what is desired.

Example 2: An example of using the **sshd** keyword ChrootDirectory and the "internal-sftp" subsystem for a specific group of users. Users who are members of the group SFTPUSERS will have their root directory set to "/files/repository" and be forced into using **sftp**, regardless of the command they are attempting to run. If they are not members, their root directory will not be changed when connecting.

They will also not be limited to only using **sftp** unless other **sshd** keywords were in effect for those users, such as a different ForceCommand in another Match block.

Server (name is "server1") **sshd_config** keywords:

```
Subsystem sftp internal-sftp
Match group SFTPUSERS
   ChrootDirectory /files/repository
   ForceCommand internal-sftp
```

Client (user "employee1" in group SFTPUSERS, home directory is /u/employee1):

```
> sftp server1
 Connecting to server1...
 sftp> pwd
 Remote working directory: /
 sftp> ls -a
 ...          file1        file2
```

Configuring the system for X11 forwarding

X11 forwarding allows users who have an account on a UNIX machine to open a connection to the X11 interface remotely from another computer. Because this connection uses SSH, the communication between the systems is encrypted. X11 forwarding works only if the system being connected to has both SSH and X11 forwarding enabled.

Guideline: Enable X11 forwarding with caution. Users with the ability to bypass file permissions on the remote host (for the user's X authorization database) can access the local X11 display through the forwarded connection. Unauthorized users might then be able to perform activities such as keystroke monitoring.

Steps for configuring the system for X11 forwarding

Before you begin: You need to know what local directory you want to copy the files from /usr/lpp/tcpip/X11R6/Xamples/clients/xauth to.

Perform the following steps to configure your system for X11 forwarding. The first two steps explain how to install the xauth sample program.

1. Copy the files from the /usr/lpp/tcpip/X11R6/Xamples/clients/xauth directory to a local directory.

   Example: Copy the files from the /usr/lpp/tcpip/X11R6/Xamples/clients/xauth directory to the local directory /u/Billy/XauthBuild.

   ```
   cp -R /usr/lpp/tcpip/X11R6/Xamples/clients/xauth  /u/Billy/XauthBuild
   ```

2. Edit the Makefile in your copied directory.

   a. Change CFLAGS to:

      ```
      CFLAGS = -D_ALL_SOURCE -DTCPCONN -DUNIXCONN -I/usr/lpp/tcpip/X11R6/include
      ```

   b. Change SYSLIBS to:

      ```
      SYSLIBS =  -lXaw -lXmu -lXt -lSM -lICE -lXext -lX11 -lXau
      ```

      These changes enable the xauth program to run without using DLLs. If you want xauth to use DLLs, enable the PermitUserEnvironment **sshd** configuration option so that LIBPATH can be read from the user's

environment file. However, because enabling might allow users to bypass access restrictions, enabling it is not recommended.

c. Compile the code by issuing **make**. You will need the _C89_CCMODE environment variable set. To enable it only for this command invocation, issue **make** as follows:

```
_C89_CCMODE=1 make
```

d. Move the xauth binary to the desired installation location.

---

3. Configure the server for X11 forwarding.

   a. Verify that the **sshd** configuration variable UseLogin is disabled. It is disabled by default.

   b. Change the **sshd** configuration variable X11Forwarding to "yes".

   c. Verify that the **sshd** configuration variable X11UseLocalhost is set to "yes". (The default setting is "yes".)

   d. Set the **sshd** and **ssh** configuration variable XAuthLocation to the full path name of the new xauth executable in both the system-wide **ssh** and **sshd** configuration files. The xauth program might need to support the generate command in order to allow **ssh** to successfully set up untrusted X11 forwarding.

   Optionally, you can set X11Display Offset to a desired value.

When you are done, you have configured your system for X11 forwarding. Users will have to configure their setup for X11 forwarding, as described in “Steps for configuring your setup for X11 forwarding” on page 74.

When users cannot log in using ssh, scp or sftp

Certain setup problems or configurations might prevent a user from using **ssh**, **scp** or **sftp** to login.

Table 14. Setup and configuration problems that can prevent users from logging in using ssh, scp, or sftp

Problem	Solution
The user's files and directories are not sufficiently protected from others.	In the **sshd_config** description, see “StrictModes” on page 156 and “ChrootDirectory” on page 146.
The system administrator limited the number of concurrent connection attempts (unauthenticated users).	In the **sshd_config** description, see “MaxStartups” on page 153. The default is 10. You might want to change the MaxStartups value because 10 connection attempts at once might not be enough for your z/OS system.
The system administrator denied a particular user, group, or IP address to the system.	In the **sshd_config** description, see “AllowUsers” on page 145, “DenyUsers” on page 149, “AllowGroups” on page 145, and “DenyGroups” on page 148. In the **sshd** description, see “from=pattern-list” on page 121. In the **sshd** description, see “/etc/nologin” on page 125. In the **sshd_config** description, see “MaxAuthTries” on page 153.
The user waited too long to enter the password.	In the **sshd_config** description, see “LoginGraceTime” on page 152.

Table 14. Setup and configuration problems that can prevent users from logging in using ssh, scp, or sftp (continued)

Problem	Solution
The user is trying to use a certain authentication method but is failing.	The system administrator might have disabled that authentication method. See "sshd_config" on page 144.
The user has an incorrect public host key in the known_hosts file.	Verify the public host key for the remote host, and update the known_hosts file.

Using hardware support to generate random numbers

If Integrated Cryptographic Service Facility (ICSF) is available, OpenSSH uses hardware support (`/dev/random` or `/dev/urandom`) to generate random numbers instead of using the OpenSSH software algorithm **ssh-rand-helper**. This improvement eliminates any timeout issues that might occur while using **ssh-rand-helper**.

OpenSSH checks for the hardware support (`/dev/random` or `/dev/urandom`) first and will use the hardware support if it is available. If ICSF is not available or if `/dev/random` and `/dev/urandom` are not available, OpenSSH reverts to using **ssh-rand-helper**. For more information about ICSF, see *z/OS Cryptographic Services ICSF Overview*.

Rule: In order for OpenSSH to use the hardware support (`/dev/random` or `/dev/urandom`) to collect random numbers, the ICSF started task must be running and the user ID must have READ access to the CSFRNG (random number generate service) profile in the RACF CSFSERV class. If the user ID does not have READ access to the CSFRNG profile, a RACF warning is issued on the MVS console.

Example: A warning for user WELLIE1 would look like the following output:

```
ICH408I USER(WELLIE1   ) GROUP(SYS1   ) NAME(WELLIE1)
 CSFRNG CL(CSFSERV )
 INSUFFICIENT ACCESS AUTHORITY
 FROM CSFRNG (G)
 ACCESS INTENT(READ)  ACCESS ALLOWED(NONE)
```

Steps for authorizing users to the random number generate service (CSFRNG)

Before you begin: You need to be sure that the CSFRNG resource profile has been defined. If it hasn't, then issue the following command where CSFSERV is the class name and CSFRNG is the profile name:

```
RDEFINE CSFSERV CSFRNG UACC(NONE)
```

Perform the following steps to authorize users to the random number generate service (CSFRNG):

1. Use one of the following commands to give READ access to the CSFRNG profile, based on your site's security policy:
   - To give a user READ access to the CSFRNG profile, where *userid* is the UID for the specified user, issue:

     ```
     PERMIT CSFRNG CLASS(CSFSERV) ID(userid) ACCESS(READ)
     ```

     If you choose to give READ access to individual users, you need to repeat this step for each user who requires access.

- To give READ access for a specific group to the CSFRNG profile where *groupid* is the GID for the specified group, issue:

  ```
  PERMIT CSFRNG CLASS(CSFSERV) ID(groupid) ACCESS(READ)
  ```

  Verify that the intended user IDs are added to the group.
- To give READ access for all RACF-defined users and groups to the CSFRNG profile, issue:

  ```
  PERMIT CSFRNG CLASS(CSFSERV) ID(*) ACCESS(READ)
  ```

  Giving all users and groups READ access to the CSFRNG profile is an unconditional way to authorize users. The security administrator must take the site's security policy into consideration when deciding whether to give all RACF-defined users and groups access to CSFRNG. *z/OS Cryptographic Services ICSF Administrator's Guide* has information about the CSFRNG profile.

---

2. Verify that all user IDs given access to this class have an OMVS segment defined and are not using the default OMVS segment.

---

3. Refresh the CSFSERV class.

   ```
   SETROPTS RACLIST(CSFSERV) REFRESH
   ```

When you are done, you have authorized users to the random number generate service (CSFRNG).

Verifying if hardware support is being used

The simplest way to verify if OpenSSH is using hardware support (`/dev/random` or `/dev/urandom`) to collect random numbers, is to start **ssh** in debug mode.

- If the debug statement shows "Seeding PRNG from /usr/lib/ssh/ssh-rand-helper", then the software algorithm **ssh-rand-helper** was used.

  Example:

  ```
  > ssh -vvv user@host
  ```

  Result:

  ```
  OpenSSH_5.0p1, OpenSSL 0.9.8k 25 Mar 2009
  debug1: Reading configuration data /etc/ssh/ssh_config
  debug3: Seeding PRNG from /usr/lib/ssh/ssh-rand-helper
  ```

- If the debug statement shows "RNG is ready, skipping seeding", then hardware support (`/dev/random` or `/dev/urandom`) was used.

  Example:

  ```
  > ssh -vvv user@host
  ```

  Result:

  ```
  OpenSSH_5.0p1, OpenSSL 0.9.8k 25 Mar 2009
  debug1: Reading configuration data /etc/ssh/ssh_config
  debug3: RNG is ready, skipping seeding
  ```

Setting up OpenSSH to collect SMF records

You can set up the system and OpenSSH to collect SMF Type 119 records for both the client and the server.

Steps for setting up the system to collect OpenSSH SMF records

Perform the following steps to set up the system to collect OpenSSH SMF records.

1. Update the SMFPRMxx parmlib member to activate SMF data collection for Type 119 and subtype 96, 97, and 98 records. For example:

   ```
   SYS(TYPE(119(96:98)))
   ```

   ---

2. Update the SMFPRMxx parmlib member to indicate which SMF exits (IEFU83 or IEFU84) are desired. For example:

   ```
   SYS(EXITS(IEFU83,IEFU84))
   ```

When you are done, you have set up the system to collect SMF records. For more information, see:

- *z/OS MVS System Management Facilities (SMF)*
- *z/OS MVS Initialization and Tuning Reference*

Steps for setting up OpenSSH to collect SMF records

Before you begin: You need to make sure that the system has been set up to collect OpenSSH SMF records as described in "Steps for setting up the system to collect OpenSSH SMF records." You also need to ensure that you have done the steps listed in "What you need to verify before using OpenSSH" on page 21.

Perform the following steps to set up OpenSSH to collect SMF records.

1. To enable SMF recording for the client side, in the `/etc/ssh/zos_ssh_config` file, set the keyword:

   ```
   ClientSMF    TYPE119_U83
   ```

   or

   ```
   ClientSMF    TYPE119_U84
   ```

   Restriction: The ClientSMF keyword can only be set in the z/OS-specific system-wide OpenSSH client configuration file. See "zos_ssh_config" on page 141 for more information.

   ---

2. To enable SMF recording for the server side, in the `/etc/ssh/zos_sshd_config` file, set the keyword:

   ```
   ServerSMF    TYPE119_U83
   ```

   or

   ```
   ServerSMF    TYPE119_U84
   ```

   Restriction: The ServerSMF keyword can only be set in the z/OS-specific OpenSSH daemon configuration file. See "zos_sshd_config" on page 158 for more information.

When you are done, you have set up OpenSSH to collect SMF records.

Chapter 6. Security topics when using key rings for key management

This topic discusses security topics in connection with key rings. OpenSSH can be configured to support keys in both UNIX files and key rings for both server and user authentication.

Choosing between UNIX files and key rings

Using UNIX files to store the keys is the common method supported on all OpenSSH implementations. Consider what other OpenSSH hosts you will be communicating with; that is, are they z/OS or non-z/OS? Also consider whether the z/OS systems are using key rings.

On the other hand, key rings provide commonality with other z/OS products that store keys in the security product. They can be real or virtual key rings. To use SAF key rings, you must have RACF or an alternative security product with compatible support. Authority must also be given to user IDs to manage the key rings. For more information about key rings, see *z/OS Security Server RACF Security Administrator's Guide*.

Restriction: If you are using SSH protocol version 1, you cannot use key rings to hold your keys. You must use UNIX files to hold RSA keys used for SSH protocol version 1.

Managing key rings and restricting access to them

Authorized applications use commands or system services provided by the security product to manage key rings. This documentation typically refers to RACF commands when presenting examples of how to set up key rings. If a different security product is used, consult that product's documentation to determine whether it contains compatible support. For more information about the RACF commands referred to in this documentation, the necessary authority required to use the commands, and any other options not described in this documentation, see *z/OS Security Server RACF Command Language Reference*.

To restrict access to key rings, two methods are available: global profile checking and ring-specific profile checking.

- **Ring-specific profile checking**, which has precedence over global profile checking, uses a resource with one of the following formats to provide access control to a specific key ring.
  - For real key rings: <KeyRingOwner>.<KeyRingName>.LST
  - For virtual key rings: <KeyRingOwner>.IRR_VIRTUAL_KEYRING_LST

  For more details about name restrictions and other considerations for using ring-specific profile checking, see the description of RACF authorization in the R_datalib interface section in *z/OS Security Server RACF Callable Services*.
- **Global profile checking** uses the IRR.DIGTCERT.LISTRING resource in the FACILITY class and applies to all key rings.

Guideline: Global profile checking applies to all key rings. Ring-specific profile checking applies to a specific key ring. Ring-specific checking has precedence over

global profile checking. The method that is chosen must work with the methods of permitting and securing access to other key rings being used for OpenSSH key management or other key ring usage on your system. Because of the wide scope of coverage that global profile checking provides, ring-specific profile checking is typically the more appropriate method to use.

Validating certificates when using key rings

Each time a certificate is accessed to retrieve a public or private key, OpenSSH asks System SSL to validate the certificate first. Some of the checks performed on the certificate and all certificates in the certification chain include verifying that the current time is within the validity period, checking that the certificate is not revoked, and ensuring that the certification chain leads to a certificate obtained from a trusted data source. For a complete list of the items being validated, see the usage information for the **gsk_validate_certificate** system call in *z/OS Cryptographic Services System SSL Programming*

Although the examples used in this book do not demonstrate using root and intermediate certificate authority (CA) certificates, they are supported in the certification chain of certificates used by z/OS OpenSSH key ring support. OpenSSH treats the key ring as a trusted certificate source. Because of this, for OpenSSH to successfully validate the certification chain, all certificates in the chain must be connected to the same key ring as the end entity certificate.

Chapter 7. Globalization on z/OS systems

This topic discusses globalization on z/OS systems and the changes that must be made in order for OpenSSH to fit the globalization model.

Setting up for globalization on z/OS systems

Setting up your system or user environment for globalization on z/OS systems is a little different from what most users are accustomed to when setting up globalization on ASCII platforms. On z/OS systems, an extra step is typically needed when changing the locale. This step involves setting the character set conversion for the controlling terminal to use the correct ASCII and EBCDIC coded character sets. This action is necessary because most PC terminal emulators require ASCII data, but the z/OS shells use EBCDIC data.

For example, when using a PC emulator to interactively log into an ASCII UNIX operating system, a user will:

- On the PC, change the emulator's coded character set to match the coded character set of the remote session's locale.
- In the UNIX shell, assign the environment variable LC_ALL to a new locale, where the ASCII coded character set of that locale matches the emulator's setting.

When interactively logging into an EBCDIC z/OS UNIX operating system, the user will:

- On the PC, change the emulator's coded character set to match the ASCII coded character set of the remote session's locale. For example, the user might change the translation settings in their emulator to use coded character set ISO/IEC 8859-2 (Latin-2).
- In the UNIX shell:
  - Assign the environment variable LC_ALL to a new locale, whose EBCDIC coded character set is compatible with the ASCII coded character set used in the emulator. To determine if a coded character set is compatible with a particular locale, refer to the section in *z/OS XL C/C++ Programming Guide* that discusses locales supplied with z/OS XL C/C++.

    For example, a user might issue:

    ```
    export LC_ALL=Hu_HU.IBM-1165
    ```

    LC_ALL can be assigned after making the **ssh** connection by using the SendEnv **ssh** keyword to send the client's LC_ALL environment variable to the server. The server must be configured to accept this variable using the AcceptEnv **sshd** keyword. Before using this support, the client's LC_ALL variable must be set to a locale that is a valid locale name on the z/OS server.

    Refer to the descriptions of the **ssh_config** keyword "SendEnv" on page 138 and the **sshd_config** keyword "AcceptEnv" on page 144 for more information about these options.
  - If a terminal type (tty) is allocated, issue the **chcp** command to assign the EBCDIC and ASCII coded character sets, as appropriate. The specified ASCII coded character set should match that of the client emulator's setting.

    For example, a user might issue:

    ```
    chcp -a ISO8859-2 -e IBM-1165
    ```

On z/OS systems, in daemons such as **rlogind**, **telnetd**, and **sshd**, conversion between ASCII and EBCDIC occurs in the forked daemon process which handles the user's connection. This process allocates the terminal (tty) for the end user. On ASCII platforms, no conversion is necessary.

OpenSSH and globalization

OpenSSH assumes that all text data traveling across the network is encoded in ISO/IEC 8859-1 (Latin-1). Specifically, OpenSSH treats data as text and performs conversion between the ASCII Latin-1 coded character set and the EBCDIC-coded character set of the current locale in the following scenarios:

- **ssh** login session
- **ssh** remote command execution
- **scp** file transfers
- **sftp** file transfers when the ascii subcommand is specified

The OpenSSH daemon (**sshd**) can understand and handle non-Latin-1 coded character sets on the network for interactive sessions, specifically sessions with a tty allocated. However, not all EBCDIC-coded character sets are compatible with ISO 8859-1. To determine if a coded character set is compatible with a particular locale, see the information about locales supplied with z/OS XL C/C++ in *z/OS XL C/C++ Programming Guide*.

Warning: If there is no one-to-one mapping between the EBCDIC coded character set of the session data and ISO 8859-1, then nonidentical conversions might occur. Specifically, substitution characters (for example, IBM-1047 0x3F) are inserted into the data stream for those incompatible characters. See "Configuring the OpenSSH daemon" on page 57 and "Configuring the OpenSSH client" on page 57 for more information.

Sessions that are considered interactive include:

- The **ssh** login session when a tty is allocated. This is the default behavior.
- The **ssh** remote command execution, when the **-t** option is used to allocate a tty.

The following scenarios are considered noninteractive and continue to interpret network data as ISO 8859-1:

- The **ssh** login session when the **-T** option is specified (which disables tty allocation.)
- The **ssh** remote command execution when the **-t** option is not specified. The default behavior is not to allocate a tty for remote command execution.
- The **scp** file transfers
- The **sftp** file transfers when the ascii subcommand is specified

The support provided by IBM Ported Tools for z/OS: OpenSSH is summarized in Table 15 on page 57. It lists the expected coded character set for the network data during both interactive and noninteractive OpenSSH sessions with various peers.

Table 15. Summary of support provided by OpenSSH V1R2

Scenario	Session is:	Client is running:	Server is running:	Coded character set of network data is:
1	Interactive	z/OS	z/OS	ASCII coded character set as defined by the **chcp** setting. **Restriction:** The z/OS client expects Latin-1, so the ASCII coded character set must be handled accordingly on the server side. See "Configuring the OpenSSH daemon" for more information.
2	Interactive	Non-z/OS UNIX (such as AIX®, Linux®) or PC	z/OS	ASCII coded character set as defined by the **chcp** setting.
3	Interactive	z/OS	Non-z/OS UNIX (such as AIX, Linux) or PC	ISO 8859-1
4	Noninteractive	z/OS	z/OS	ISO 8859-1
5	Noninteractive	Non-z/OS UNIX (such as AIX, Linux) or PC	z/OS	ISO 8859-1
6	Noninteractive	z/OS	Non-z/OS UNIX (such as AIX, Linux) or PC	ISO 8859-1

Note that some OpenSSH sessions transfer data as binary. In other words, no character translation is performed. These include:

- **sftp** sessions (when the ascii subcommand is not used)
- Port-forwarded sessions
- X11-forwarded sessions

Limitation: IBM Ported Tools for z/OS: OpenSSH does not support multibyte locales.

Configuring the OpenSSH daemon

The OpenSSH daemon (**sshd**) must be run in the POSIX C locale. In most cases, this occurs without any action on behalf of the user. However, an alternate locale could inadvertently be picked up through the shell profile of the user ID invoking the daemon, or through the ENVAR run-time option in CEEPRMxx member of SYS1.PARMLIB. You can enforce LC_ALL=C by using STDENV in the BPXBATCH job that starts the daemon.

For more information about the POSIX C locale, see *z/OS XL C/C++ Programming Guide*.

Configuring the OpenSSH client

The OpenSSH daemon (**sshd**) can understand and handle non-Latin-1 coded character sets for interactive sessions, specifically those with a tty allocated. However, the OpenSSH client (**ssh**) still expects network data to be encoded in ISO 8859-1.

If the EBCDIC coded character set for your sessions is compatible with ISO 8859-1, the following setup is not required. To determine if a coded character set is compatible with a particular locale, refer to the section on locales supplied with z/OS XL C/C++ in *z/OS XL C/C++ Programming Guide*.

If **chcp** is issued in your environment, verify that the SSH peer supports the specified ASCII coded character set.

For example, if you are using a PC to connect directly to z/OS, you issue the **chcp** command in the remote z/OS shell to assign the ASCII-coded character set for the terminal to match that of the PC emulator. The daemon inherits the **chcp** setting to translate the network data accordingly. The SSH peer, the PC emulator, must also support the new ASCII coded character set. This can be determined by checking your emulator's configuration.

If you are issuing the **ssh** client from z/OS to connect to a z/OS platform running in another locale, you need to verify that the ASCII coded character set of the remote session (set by **chcp**) is ISO 8859-1, which is what the z/OS **ssh** client expects.

Warning: If there is no one-to-one mapping between the EBCDIC coded character set of the session data and ISO 8859-1, then nonidentical conversions might occur. Specifically, substitution characters (for example, IBM-1047 0x3F) may be inserted into the data stream for those incompatible characters.

If the EBCDIC coded character set of your target locale is not compatible with ISO 8859-1, then nonidentical conversions may occur in either of these scenarios:
- You are running in the target locale when issuing the **ssh** command locally.
- You are running in the target locale in your remote **ssh** session.

To avoid nonidentical conversions, you can force the **ssh** client process to run in the C locale. Note also that the remote session's shell must also be configured to run in either the C locale or a locale with a coded character set that is compatible with ISO 8859-1.

To force the local **ssh** client process to run in a C locale, you can run **ssh** as follows:

```
LC_ALL=C ssh [arguments]
```

where arguments represents the remainder of the arguments passed to **ssh**.

You can set up a shell alias to avoid repeatedly typing the above command. For example:

```
alias ssh="LC_ALL=C ssh"
```

Configuring ssh when LC_ALL is set through shell profiles

If all the following are true for your environment:
- Your system is configured to run in a locale other than the default C locale
- The corresponding ASCII coded character set for your locale is **not** ISO 8859-1
- You changed the system-wide locale by setting LC_ALL through shell profiles (for example, `/etc/profile` or `$HOME/.profile`.)

then perform the following steps as part of your OpenSSH system-wide setup.

If you have changed the locale at a system-wide level, consider defining this alias in an area where it can be picked up by all users and inherited by all subshells. Shell aliases are typically defined through the file named by the ENV variable of `/bin/sh`. Users may have defined their own ENV setting in one of their shell profiles. For this setup, the ENV variable should be exported so it is inherited by subshells.

- For `/bin/sh` users, this alias should be defined in the ENV file.
- For `/bin/tcsh` users, this alias should be defined in `/etc/csh.cshrc`.

Steps to follow for setting up a system-wide alias for ssh

The steps assume that you are using the `/bin/sh` shell.

1. Create a UNIX file `/etc/ssh/.sshalias` that contains the following line:

   ```
   alias ssh="LC_ALL=C ssh"
   ```

2. Ensure that the UNIX permissions for this file are world-readable. From the UNIX prompt, issue:

   ```
   chmod 744 /etc/ssh/.sshalias
   ```

3. Notify users to either add the **ssh** alias to their ENV file or read in the above ENV file from their user-defined ENV file. For example, users can add to their ENV file the following line, which reads in (or "sources") the new **ssh** alias file using the **dot** command:

   ```
   . /etc/ssh/.sshalias
   ```

4. Verify that the **ssh** alias is set properly. From a ***new*** UNIX shell, issue:

   ```
   > alias ssh
   ssh="LC_ALL=C ssh"
   >
   ```

Configuring ssh when LC_ALL is set through the ENVAR run-time option in CEEPRMxx

If all the following statements are true for your environment

- Your system is configured to run in a locale other than the default C locale
- The corresponding ASCII code page for your locale is not ISO 8859-1
- You changed the system-wide locale by setting LC_ALL through the ENVAR run-time option in a CEEPRMxx member of SYS1.PARMLIB or through the operator command SETCEE.
  - For information about SETCEE, see *z/OS MVS System Commands*.
  - *z/OS MVS Initialization and Tuning Reference* contains information about the ENVAR run-time option for CEEPRMxx.

then perform the following steps as part of your OpenSSH system-wide setup.

Create an alias for the **ssh** command which forces **ssh** to run in a C locale. This alias should be defined in an area where it will be picked up by all users and all subshells, even when a login shell is not used. Shell aliases are typically defined through the file named by the ENV variable of `/bin/sh`. The ENVAR run-time option in CEEPRMxx can also be used to set a shell alias.

Steps to follow for setting up a system-wide alias for ssh through the ENVAR run-time option of CEEPRMxx

1. Create a UNIX file `/etc/ssh/.sshalias` which contains the following line:

   ```
   alias ssh="LC_ALL=C ssh"
   ```

2. Ensure that the UNIX permissions for this file are world-readable. From the UNIX prompt, issue:

```
chmod 744 /etc/ssh/.sshalias
```

3. Notify users to define this alias if they already have created their own ENV file. Users might have defined their own ENV setting in one of their shell profiles. Their ENV setting is not inherited for remote command execution or remote **ssh** processes, because these are not login shells. However, ENV will be initialized to their own setting for interactive shells, where users might later be issuing the **ssh** command. Their ENV setting overrides the ENVAR setting through CEEPRMxx, so they need to pick up your alias for local **ssh** command invocations.
   - For `/bin/sh` users, this alias should be defined in the file specified by the ENV variable.
   - For `/bin/tcsh` users, this alias should be defined in `/etc/csh.cshrc`.

   The subsequent examples all assume that one is working with `/bin/sh` users.

   Notify users to either add the **ssh** alias to their ENV file or read in your ENV file from their ENV file. For example, users might add to their ENV file the following line, which reads in (or "sources") the new **ssh** alias file using the **dot** command:

   ```
   .  /etc/ssh/.sshalias
   ```

4. Issue the operator command SETCEE to change the CEEPRMxx setting dynamically. For example:

   ```
   SETCEE  CEEDOPT,ENVAR('LC_ALL=Hu_HU.IBM-1165','ENV=/etc/ssh/.sshalias')
   ```

5. Verify that the **ssh** alias is set properly. From a new UNIX shell, issue:

   ```
   > echo $ENV
   /etc/ssh/.sshalias
   > alias ssh
   ssh="LC_ALL=C ssh"
   >
   ```

Configuring sftp

By default, **sftp** treats files as binary. Use sftp if you do not want your data files altered. If you want your data files translated between ASCII and EBCDIC, use **iconv** to convert the files at the start or end of the **sftp** transfer.

If you have existing sftp jobs that use the ascii sftp subcommand: The ascii **sftp** subcommand converts between ASCII ISO 8859-1 and the EBCDIC of the current locale. If the file data on the network is in a coded character set that is not ISO 8859-1, then you must adjust existing jobs to transfer files as binary and use **iconv** for the data conversion.

Configuring scp

By default, **scp** treats files as text. It assumes that all data going over the network is encoded in ASCII coded character set ISO 8859-1. The EBCDIC coded character set of the current locale is used for data conversion. On the remote system, the locale of the **scp** process is determined by how LC_ALL is initialized on that system. If LC_ALL is set through a shell profile (for example, `/etc/profile`), then it will not be inherited by the remote **scp** process. Specifically, the remote **scp** process will run in a C locale. Figure 3 on page 61 shows the change in locales; for example, if a user on Host GERMANY running in locale De_DE.IBM-273 uses **scp** to transfer a file to a remote host, the file contents are converted from IBM-273 to ISO 8859-1 to go over the network and from ISO 8859-1 to IBM-1047 on the target system.

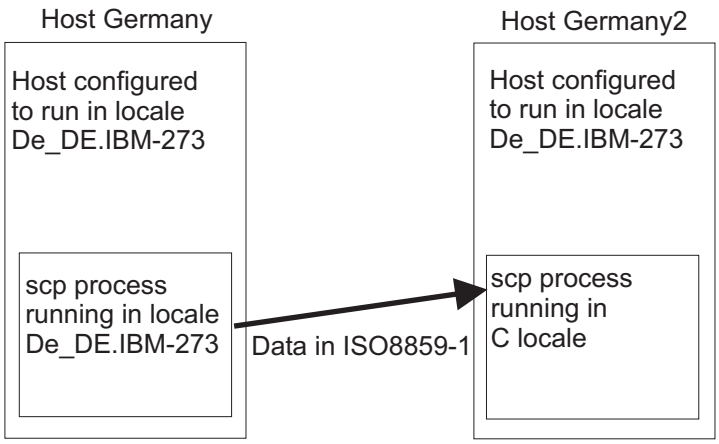

Figure 3. Using scp when LC_ALL is set through shell profiles

If LC_ALL is set through the ENVAR run-time option in the CEEPRMxx member, then the new locale is inherited by the remote **scp** process. Specifically, the EBCDIC coded character set of that locale is used. See Figure 4 for an example of using **scp** when LC_ALL is set through ENV in CEEPRMxx. If a user on Host GERMANY running in locale De_DE.IBM-273 uses **scp** to transfer a file to a remote host, the file contents are converted from IBM-273 to ISO 8859-1 to go over the network, and from ISO 8859-1 to IBM-273 on the target system.

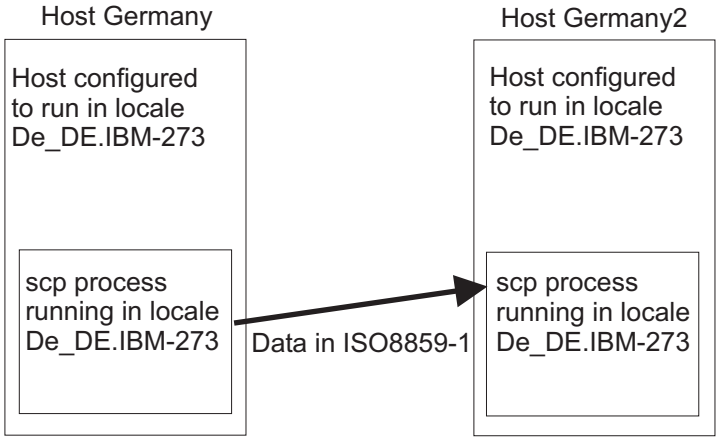

Figure 4. Using scp when LC_ALL is set through ENV in CEEPRMxx

Warning: If a file is encoded in an EBCDIC coded character set whose compatible ASCII coded character set is not ISO 8859-1, then nonidentical conversions might occur. Specifically, substitution characters (for example, IBM-1047 0x3F) might replace characters that do not have a mapping between the specified EBCDIC coded character set and ISO 8859-1. To determine if a coded character set is compatible with a particular locale, see the information about locales supplied with z/OS XL C/C++ in *z/OS XL C/C++ Programming Guide*.

If the EBCDIC coded character set for your sessions is compatible with ISO 8859-1 and the preceding text conversions are satisfactory for your environment, the following setup is not required.

If you have existing scp jobs

If you are changing the locale on a system whose ASCII coded character set is not Latin-1 and you have existing **scp** jobs configured, you can:

- Convert those jobs to use **sftp**.
- Force **scp** to treat files as though they are encoded in IBM-1047, so substitution characters are not introduced. This can be done through a shell alias, as described in "Configuring scp when LC_ALL is set through shell profiles."
- If you intend to configure a new locale through a shell profile, then continue to "Configuring scp when LC_ALL is set through shell profiles."
- If you intend to configure a new locale using CEEPRMxx to specify run-time options, then continue to "Configuring scp when LC_ALL is set through the ENVAR run-time option in CEEPRMxx."

Configuring scp when LC_ALL is set through shell profiles

If all the following are true for your environment:

- Your system is configured to run in a locale other than the default C locale
- The corresponding ASCII coded character set for your locale is **not** ISO 8859-1
- You changed the system-wide locale by setting LC_ALL through shell profiles (for example, `/etc/profile` or `$HOME/.profile`.
- You do not want to convert existing **scp** workloads to **sftp** workloads

then perform the following steps as part of your OpenSSH system-wide setup.

If you have changed the locale at a system-wide level, consider defining this alias in an area where it can be picked up by all users and inherited by all subshells. Shell aliases are typically defined through the file named by the ENV variable of `/bin/sh`. Users might have defined their own ENV setting in one of their shell profiles. For this setup, the ENV variable should be exported so it is inherited by subshells.

- For `/bin/sh` users, this alias should be defined in the ENV file.
- For `/bin/tcsh` users, this alias should be defined in `/etc/csh.cshrc`.

Steps to follow for setting up a system-wide alias for scp

The steps assume that you are using the `/bin/sh` shell.

1. Create a UNIX file, `/etc/ssh/.sshalias`, that contains the following line:

   ```
   alias scp="LC_ALL=C scp"
   ```

2. Ensure that the UNIX permissions for this file are world-readable. From the UNIX prompt, issue:

   ```
   chmod 744 /etc/ssh/.sshalias
   ```

3. Notify users to either add the **scp** alias to their ENV file or read in the above ENV file from their user-defined ENV file. For example, users can add to their ENV file the following line, which reads in (or "sources") the new **scp** alias file using the **dot** command:

   ```
   . /etc/ssh/.sshalias
   ```

4. Verify that the **scp** alias is set properly. From a *new* UNIX shell, issue:

   ```
   > alias scp
   scp="LC_ALL=C scp"
   >
   ```

Configuring scp when LC_ALL is set through the ENVAR run-time option in CEEPRMxx

If all the following are true for your environment:

- Your system is configured to run in a locale other than the default C locale

- The corresponding ASCII code page for your locale is **not** ISO 8859-1
- You changed the system-wide locale by setting LC_ALL through the ENVAR run-time option in a CEEPRMxx member or through the SETCEE operator command.
  - For information about SETCEE, see *z/OS MVS System Commands*.
  - *z/OS MVS Initialization and Tuning Reference* contains information about the ENVAR run-time option for CEEPRMxx.
- You do not want to convert existing **scp** workloads to **sftp** workloads

then perform the following steps as part of your OpenSSH system-wide setup.

Steps to follow for setting up a system-wide alias for scp through the ENVAR run-time option of CEEPRMxx

1. Create a UNIX file `/etc/ssh/.sshalias` that contains the following line:

   ```
   alias scp="LC_ALL=C scp"
   ```

2. Ensure the UNIX permissions for this file are world-readable. From the UNIX prompt, issue:

   ```
   chmod 744 /etc/ssh/.sshalias
   ```

3. Notify users to define this alias if they already have created their own ENV file. Users might have defined their own ENV setting in one of their shell profiles. Their ENV setting is not inherited for remote command execution or remote **scp** processes, because these are not login shells. However, ENV is initialized to their own setting for interactive shells, where users might later be issuing the **scp** command. Their ENV setting overrides the ENVAR setting through CEEPRMxx, so they need to pick up your alias for local **scp** command invocations.

   - For `/bin/sh` users, this alias must be defined in the file specified by the ENV variable.
   - For `/bin/tcsh` users, this alias must be defined in `/etc/csh.cshrc`.

   The subsequent examples all assume that you are working with `/bin/sh` users.

   Notify users to either add the **scp** alias to their ENV file or read in your ENV file from their ENV file. For example, users can add to their ENV file the following line, which reads in (or "sources") the new **scp** alias file using the **dot** command:

   ```
   .  /etc/ssh/.sshalias
   ```

4. Issue the SETCEE operator command to change the CEEPRMxx setting dynamically. For example:

   ```
   SETCEE  CEEDOPT,ENVAR('LC_ALL=Hu_HU.IBM-1165','ENV=/etc/ssh/.sshalias')
   ```

5. Verify that the **scp** alias is set properly. From a *new* UNIX shell, issue:

   ```
   > echo $ENV
   /etc/ssh/.sshalias
   > alias scp
   scp="LC_ALL=C scp"
   >
   ```

Customizing your UNIX environment to run in another locale

To configure your UNIX environment to run in another locale, see the section on customizing for your national code page in *z/OS UNIX System Services Planning*.

Rule: All files used by OpenSSH (such as key files and configuration files) must be in the IBM-1047 coded character set, with the exception of the**rc** files (`/etc/ssh/sshrc` and `~/.ssh/rc`). The **rc** files are parsed by `/bin/sh` and should be

in the coded character set of the current locale. Do not use the `/etc/ssh/sshrc` file if there is a possibility of the users on the system running in different locales.

Warning: While it is possible to set LC_ALL through the ENVAR run-time option of the CEEPRMxx member, configuring the locale in this way might cause unexpected results. Specifically, it is possible that daemons or long-running processes might expect to run in a C locale. Verify that all these processes support running in your alternate locale. Additionally, some system administration user IDs might need to run in a C locale, for editing configuration files which expect to be encoded in IBM-1047.

Chapter 8. Getting ready to use OpenSSH

This topic discusses the setup tasks that the user must do. It includes the steps for setting up user authentication, which is a required step and also discusses how to set up the system for X11 forwarding, which is an optional step.

Requirement: All files used by OpenSSH (such as key files and configuration files) must be in the IBM-1047 code set, except for the **rc** files (`/etc/ssh/sshrc` and `~/.ssh/rc`). The **rc** files are parsed by `/bin/sh` and must be in the code set of the current locale. Do not use the `/etc/ssh/sshrc` file if users on the system might be running in different locales.

Restriction: OpenSSH does not run in multibyte locales.

Setting up the OpenSSH client configuration files

The settings in the OpenSSH client configuration files (**ssh_config** and **zos_user_ssh_config**) provide system defaults and can be overridden by command-line options.

Steps for setting up the OpenSSH client configuration files

Before you begin: You must be running in the default C locale before performing these steps.

1. Customize the OpenSSH client configuration file.
   a. Copy the sample **ssh_config** configuration file from the `/samples` directory to your `~/.ssh` directory.

      ```
      cp /samples/ssh_config ~/.ssh/config
      chmod 600 ~/.ssh/config
      ```

   b. Modify the `~/.ssh/config` file to control the SSH client-side authentication methods attempted, protocols and ciphers supported, and session control options. For details, see "ssh" on page 85 and "ssh_config" on page 129.

---

2. Customize the z/OS-specific per-user client configuration file.
   a. Copy the sample **zos_user_ssh_config** file from the `/samples` directory to the `~/.ssh` directory.

      ```
      cp /samples/zos_user_ssh_config ~/.ssh/zos_user_ssh_config
      chmod 600 ~/.ssh/zos_user_ssh_config
      ```

   b. Modify the **zos_user_ssh_config** file to control the z/OS-specific per-user client options. For details, see "ssh" on page 85 and "zos_user_ssh_config" on page 142.

When you are done, you have set up the OpenSSH client configuration files.

Setting up user authentication

Before clients can verify their identities to the server using public key authentication, user authentication must be set up first. Public key authentication is the most secure authentication method available in SSH. A user creates both a public and private key and then transfers a copy of the public key to the **ssh** server being accessed. The private key is kept on the user's local machine and is used to verify the identity of the user when the user attempts to connect to the **ssh** server. The public and private keys must be correct for the server to allow the connection. Those keys can be stored in either UNIX files or SAF key rings, or both. For more information about storing the key rings, see "Choosing between UNIX files and key rings" on page 53.

Restriction: If you are using SSH protocol version 1, you cannot use key rings to hold your keys. You must use UNIX files to hold RSA keys used for SSH protocol version 1.

The procedures for setting up user authentication are described in the following sections:

- "Steps for setting up user authentication when keys are stored in UNIX files"
- "Steps for setting up user authentication when keys are stored in key rings" on page 68

Steps for setting up user authentication when keys are stored in UNIX files

Perform the following steps to set up user authentication.

1. Generate public and private key pairs, based on the SSH protocol you plan to use, protocol version 1 or protocol version 2.

   If you are using SSH protocol version 1, issue:

   ```
   ssh-keygen -t rsa1
   ```

   If you are using SSH protocol version 2, issue:

   ```
   ssh-keygen -t rsa
   ssh-keygen -t dsa
   ```

   ---

2. On the remote host, distribute the public keys to all remote hosts that you plan to log in to, using public key authentication. By default, OpenSSH uses the `authorized_keys` file to store these public keys. Figure 5 on page 68 shows an example of the steps to follow in order to create an `authorized_keys` file when keys are stored in UNIX files.

   a. Create or edit the `~/.ssh/authorized_keys` file for your accounts on both local and remote systems.

   b. Append the public keys to the `~/.ssh/authorized_keys` file as follows:

      - To enable local users to log into a remote account, append the local user's public keys (those ending with a "pub" suffix) to the remote user's `~/.ssh/authorized_keys` file.
      - To enable remote users to log into a local account, append the remote user's public keys (those ending with a "pub" suffix) to the local user's `~/.ssh/authorized_keys` file.

You can append the public keys by using cut and paste. Because a key is a long line, make sure that the keys are not split across lines. Each key should be exactly one line of the file.

If you use FTP to copy your public key files to another system, treat the files as text to enable any necessary conversion between ASCII and EBCDIC.

---

3. On the remote host that you plan to log into, verify that your home directory (for example, ~/), the `.ssh` subdirectory, and the `authorized_keys` file are not writable by other users or their owning group. The default configuration of the OpenSSH daemon enables StrictModes, which verifies these settings before allowing public key authentication.

When you are done, you have set up user authentication. Every time you regenerate the keys, you must update the `authorized_keys` file on remote systems.

Example of user authorization when keys are stored in UNIX files

An employee named Bill has two accounts on two systems where UNIX files are used to store keys. His user name on HOST1 is BILLY. On HOST2, his user name is WILLIAM. While logged into HOST1, he wants to be able to access HOST2 using **ssh** with public key authentication. Figure 5 on page 68 shows how the process would work.

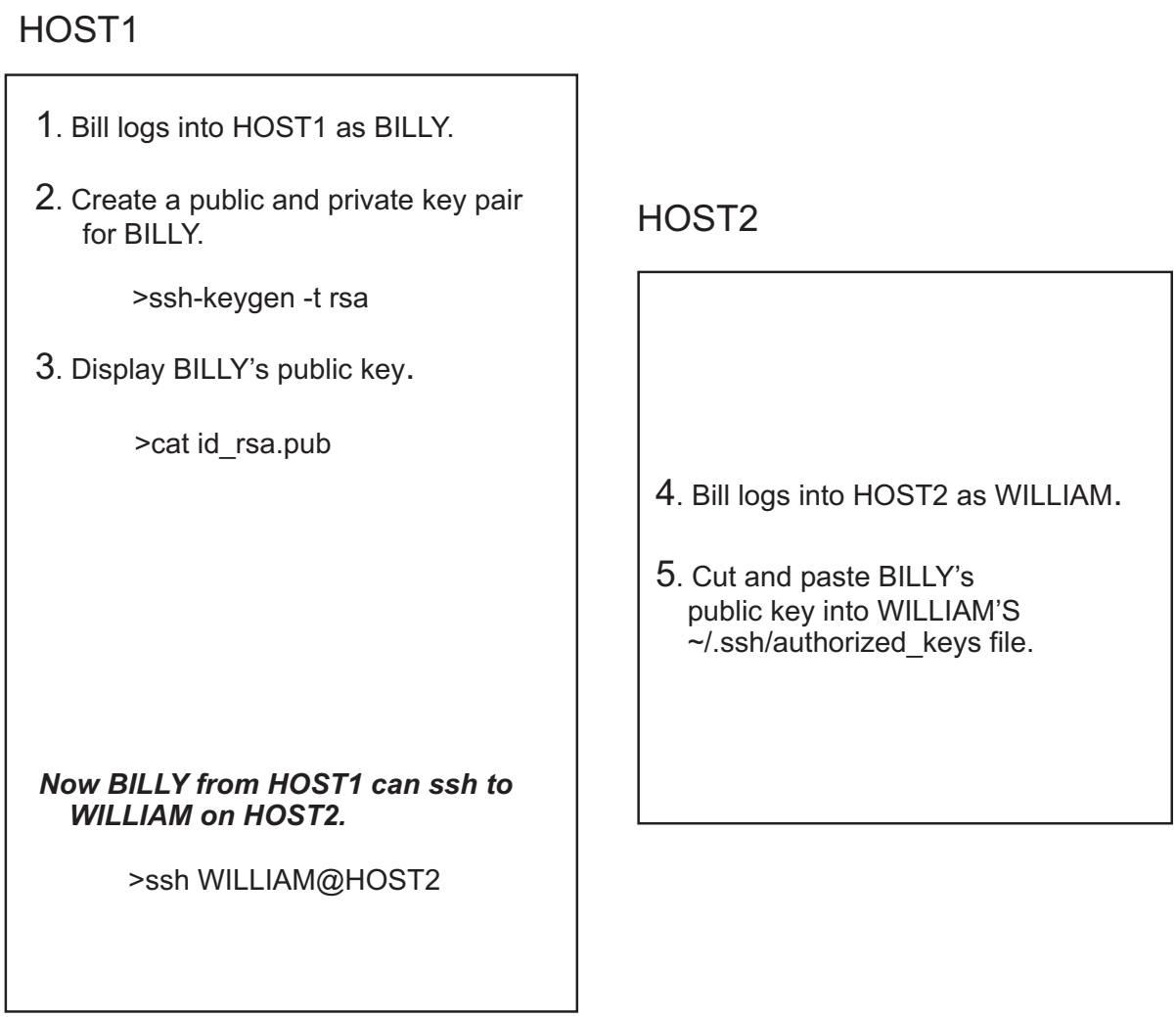

Figure 5. Accessing a remote system using ssh with public key authentication when keys are stored in UNIX files

Steps for setting up user authentication when keys are stored in key rings

The setup procedure has been divided into two steps:

- "Step 1. Construct the key ring" on page 69
- "Step 2. Distribute the public keys to all remote hosts" on page 71

Notes about the command examples

The examples for managing key rings and associated objects use the RACDCERT RACF command. If you are using a different security product, consult that product's documentation to determine if it contains compatible support. For more information about the RACDCERT command, the necessary authority required to use the command, and any other options not described, see *z/OS Security Server RACF Command Language Reference*.

In the examples, input names that are given in italics are variables, which you can choose. Some of these names in italics contain hyphen characters (-) separating portions of the name. These hyphens are variable and are not required. The names given are suggestions and are consistently used throughout the examples (for example, if you customize your own version in one step, that name will likely need to be used on other command steps as well).

The examples demonstrate using a self-signed certificate. Using a certificate chain, such as with root and intermediate certificate authority certificates, is supported. If you will be using more advanced certificate chains than the examples demonstrate, see "Validating certificates when using key rings" on page 54 for important considerations.

Step 1. Construct the key ring

In this step, you will construct a key ring, if one is needed, generate certificates, connect them to the user's key ring, and set up permission to access the key ring.

Before you begin: You need to know the following:

- Which protocol version you will be using. If you are using SSH protocol version 1, you cannot use key rings to hold your keys. You must use UNIX files to hold RSA keys used for SSH protocol version 1.
- Whether you are working with real or virtual key rings because the setup steps vary depending on the type of key ring is being used. A virtual key ring can be used instead of a real key ring when you want to treat the collection of all the certificates owned by one user ID, including the SITE and CERTAUTH reserved user IDs, as an independent key ring. The use of virtual key rings might eliminate the need to create multiple real key rings when you have several applications that use key rings and digital certificates. See *z/OS Security Server RACF Security Administrator's Guide* for more information about real and virtual key rings.

1. Create a real key ring if you do not yet have one for your keys. Omit this step if you plan to use a virtual key ring. If you already have a key ring or are using a virtual key ring, go to Step 2 on page 70. Use the RACDCERT ADDRING command to create the new key ring, specifying the owning user ID and the key ring name. The ID keyword must specify the user ID that will be authenticating with the keys within it. The key ring name can be any unique name for this user ID.

   Example: To define the SSHring key ring, issue:

   ```
   RACDCERT ADDRING(SSHring) ID(userID)
   ```

   On this command example, and all that follow, the ID() keyword can be omitted if the invoking user is the same as the authenticating user ID.

2. Using the RACDCERT GENCERT command, generate a certificate with public and private keys, based on the algorithms that are supported on the server (either RSA, DSA, or both.) For RSA keys, the minimum size is 768 bits and the maximum size is 32768 bits. Generally, 2048 bits is considered sufficient. DSA keys must be exactly 1024 bits as specified by FIPS 186-2. DSA keys larger than 1024 bits associated with certificates in a key ring are not supported by OpenSSH.

   Do not use variant characters in the label name for the certificate.

   Examples: Although the examples demonstrate how to create non-ICSF (Integrated Cryptographic Storage Facility) certificates in the RACF database, ICSF can also be used to store the certificate and associated keys for RSA only. These can be generated by software using ICSF or by hardware using a PCI Cryptographic Coprocessor (PCICC). For more information, refer to *z/OS Cryptographic Services ICSF Administrator's Guide*.

   - To generate a certificate and an RSA public/private key pair, storing the private key in the RACF database as a non-ICSF key:

     ```
     RACDCERT GENCERT SUBJECTSDN(CN('uniq-ssh-rsa-cn')) SIZE(2048)
      WITHLABEL('uniq-ssh-rsa') ID(userID)
     ```

   - To generate a certificate and a DSA public/private key pair, storing the private key in the RACF database as a non-ICSF key:

     ```
     RACDCERT GENCERT SUBJECTSDN(CN('uniq-ssh-dsa-cn')) SIZE(1024) DSA
      WITHLABEL('uniq-ssh-dsa') ID(userID)
     ```

   The SUBJECTSDN parameter offers additional customizable keywords, which are not documented in this section, that can be included in the distinguished name. The label assigned to the certificate must be unique within the RACF database.

---

3. If real key rings are being used, use the RACDCERT CONNECT command to connect the certificate to the user's key ring. Omit this step if virtual key rings are being used. If you are not the certificate owner, you must identify the user ID that owns the certificate. If you are not the key ring owner, you must identify the user ID that owns the key ring. These will typically be the same for this connect command.

   ```
   RACDCERT CONNECT(ID(userID) LABEL('uniq-ssh-type') RING(SSHring)
    USAGE(PERSONAL)) ID(userID)
   ```

---

4. Update the user's z/OS-specific per-user client configuration file (`~/.ssh/zos_user_ssh_config`) to indicate the location of the user's keys when using key rings.

   - **If real key rings are being used,** add the following line:

     ```
     IdentityKeyRingLabel "userID/SSHring uniq-ssh-type"
     ```

   - **If virtual key rings are being used,** add the following line:

     ```
     IdentityKeyRingLabel "userID/* uniq-ssh-type"
     ```

---

5. Permit access to the key ring for the user, using either ring-specific profile checking or global profile checking. These are discussed in "Managing key rings and restricting access to them" on page 53.

   Examples:

- To define individual user access to the real key ring, SSHring, using ring-specific profile checking:

  ```
  RDEFINE RDATALIB userID.SSHring.LST UACC(NONE)
  PERMIT userID.SSHring.LST CLASS(RDATALIB) ID(userID) ACCESS(READ)
  ```

  If the RDATALIB class is not yet active and RACLISTed:

  ```
  SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
  ```

  Refresh the class:

  ```
  SETROPTS RACLIST(RDATALIB) REFRESH
  ```

- To define individual user access to the virtual key ring, using ring-specific profile checking:

  ```
  RDEFINE RDATALIB userID.IRR_VIRTUAL_KEYRING_LST UACC(NONE)
  PERMIT userID.IRR_VIRTUAL_LISTRING_LST CLASS(RDATALIB) ID(userID) ACCESS(READ)
  ```

  If the RDATALIB class is not yet active and RACLISTed:

  ```
  SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
  ```

  Refresh the class:

  ```
  SETROPTS RACLIST(RDATALIB) REFRESH
  ```

- To define individual user access, using global profile checking:

  ```
  RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(READ)
  ```

  If the FACILITY class is not yet active and RACLISTed:

  ```
  SETROPTS RACLIST(FACILITY) CLASSACT(FACILITY)
  ```

  Refresh the class:

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

Step 2. Distribute the public keys to all remote hosts

In this step, you will distribute the public keys to all remote hosts that you plan to log in to, using public key authentication. Figure 6 on page 74 shows an example of the steps to follow in order to create an `authorized_keys` file when keys are stored in key rings.

1. Export the public keys to remote hosts that store user's keys in a UNIX file (the `authorized_keys` file).
   - On the local host, use **ssh-keygen -e** to export the public key into a UNIX file.

     Example:

     ```
     _ZOS_SSH_KEY_RING_LABEL="userID/SSHring uniq-ssh-type" ssh-keygen -e > uniq-ssh.type
     ```

   - Use FTP to distribute the `uniq-ssh.type` file to the remote host.
   - On the remote host, use **ssh-keygen -i** to import the public key, appending it to the `authorized_keys` file:

     ```
     ssh-keygen -i -f uniq-ssh.type >> ~/.ssh/authorized_keys
     ```

     You have now completed distribution of the public keys to remote hosts that store user keys in a UNIX file. If you have other remote hosts that store user keys in key rings, then continue to the next step to export the public keys to remote hosts. Otherwise, you have completed Step 2.
2. Export the public keys to remote hosts that store users's keys in a certificate associated with a key ring. First, the public keys must be exported from the certificate. The RACDCERT EXPORT command can perform this type of

export. Specify the certificate identification and request CERTDER for the export format. Choose a data set to store the exported certificate and specify it on the DSN parameter. If the data set specified for DSN already exists, it is deleted and reallocated by the RACDCERT EXPORT command.

If the public key will be stored in a certificate associated with a key ring on the remote host, then export the certificate in DER format (without the private key) into a data set for each public key that needs to be distributed to remote hosts.

Example:

```
RACDCERT EXPORT(LABEL('uniq-ssh-type')) ID(userID)
 FORMAT(CERTDER) DSN('userid.sshcert.type')
```

---

3. Use FTP to distribute the exported certificate data set in binary format to the remote hosts.

---

4. On the remote host, create a real key ring if you do not yet have one for your keys. Omit this step if you plan to use a virtual key ring.

```
RACDCERT ID(userID) ADDRING(SSHAuthKeysRing)
```

---

5. On the remote host, add each user certificate into the user's SAF database.

   The RACDCERT ADD command can be used to add the exported certificate on the remote host. Specify the data set that you copied to the remote host using FTP, the user ID that should own the certificate, and indicate that this certificate is trusted. The specified user ID must be the user ID that you want to be able to connect to from the local host with the matching key. You will specify the label for this certificate on this remote host. This label must be unique for the user ID within the RACF database, and is used to identify this certificate on future commands and in authorized key files.

   This certificate only contains the public key.

   Example:

```
RACDCERT ADD('userid.sshcert.type') ID(userID)
 WITHLABEL('uniq-ssh-type') TRUST
```

---

6. On the remote host, connect each certificate to the user's key ring.

   The RACDCERT CONNECT command can be used to connect each certificate to the user's key ring if real key rings are being used. Omit this step if virtual key rings are being used. You must identify both the user ID that owns the certificate and the user ID that owns the key ring. These will typically be the same for this connect command.

   Example:

```
RACDCERT CONNECT(ID(userID) LABEL('uniq-ssh-type')
 RING(SSHAuthKeysRing) USAGE(PERSONAL)) ID(userID)
```

---

7. On the remote host, edit the `authorized_keys` file to add one line containing the *zos-key-ring-label* option for each public key that was added to the key ring. (See "Format of the authorized_keys file" on page 120 in the **sshd** command section for more information.)

   Examples:

   - **If a real key ring is being used,** add the following line:

```
zos-key-ring-label="userID/SSHAuthKeysRing uniq-ssh-type"
```

- **If a virtual key ring is being used,** add the following line:

  ```
  zos-key-ring-label="userID/* uniq-ssh-type"
  ```

---

8. On the remote host, permit access to this key ring for the user. There are two ways to provide access: ring-specific profile checking and global profile checking. Both are discussed in "Managing key rings and restricting access to them" on page 53.

   Examples:

   - To define individual user access to the real key ring, SSHAuthKeysRing, using ring-specific profile checking:

     ```
     RDEFINE RDATALIB userID.SSHAuthKeysRing.LST UACC(NONE)
     PERMIT userID.SSHAuthKeysRing.LST CLASS(RDATALIB) ID(userID) ACCESS(READ)
     ```

     If the RDATALIB class is not yet active and RACLISTed:

     ```
     SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
     ```

     Refresh the class:

     ```
     SETROPTS RACLIST(RDATALIB) REFRESH
     ```

   - To define individual user access to the virtual key ring, using ring-specific profile checking:

     ```
     RDEFINE RDATALIB userID.IRR_VIRTUAL_KEYRING_LST UACC(NONE)
     PERMIT userID.IRR_VIRTUAL_KEYRING_LST CLASS(RDATALIB) ID(userID) ACCESS(READ)
     ```

     If the RDATALIB class is not yet active and RACLISTed:

     ```
     SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
     ```

     Refresh the class:

     ```
     SETROPTS RACLIST(RDATALIB) REFRESH
     ```

   - To define individual user access, using global profile checking:

     ```
     RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(READ)
     ```

     If the FACILITY class is not yet active and RACLISTed:

     ```
     SETROPTS RACLIST(FACILITY) CLASSACT(FACILITY)
     ```

     Refresh the class:

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

When you are done, you have set up user authentication when using key rings to store keys. Every time the user keys are regenerated in the key ring, they must be redistributed and added to the key ring on the remote systems that contain the authorized keys.

Example of user authentication when keys are stored in real key rings

An employee named Bill has two accounts on two systems where real key rings are used to store keys. His user name on HOST1 is BILLY. On HOST2, his user name is WILLIAM. While logged into HOST1, he wants to be able to access HOST2 using **ssh** with public key authentication. Figure 6 on page 74 shows how the process would work.

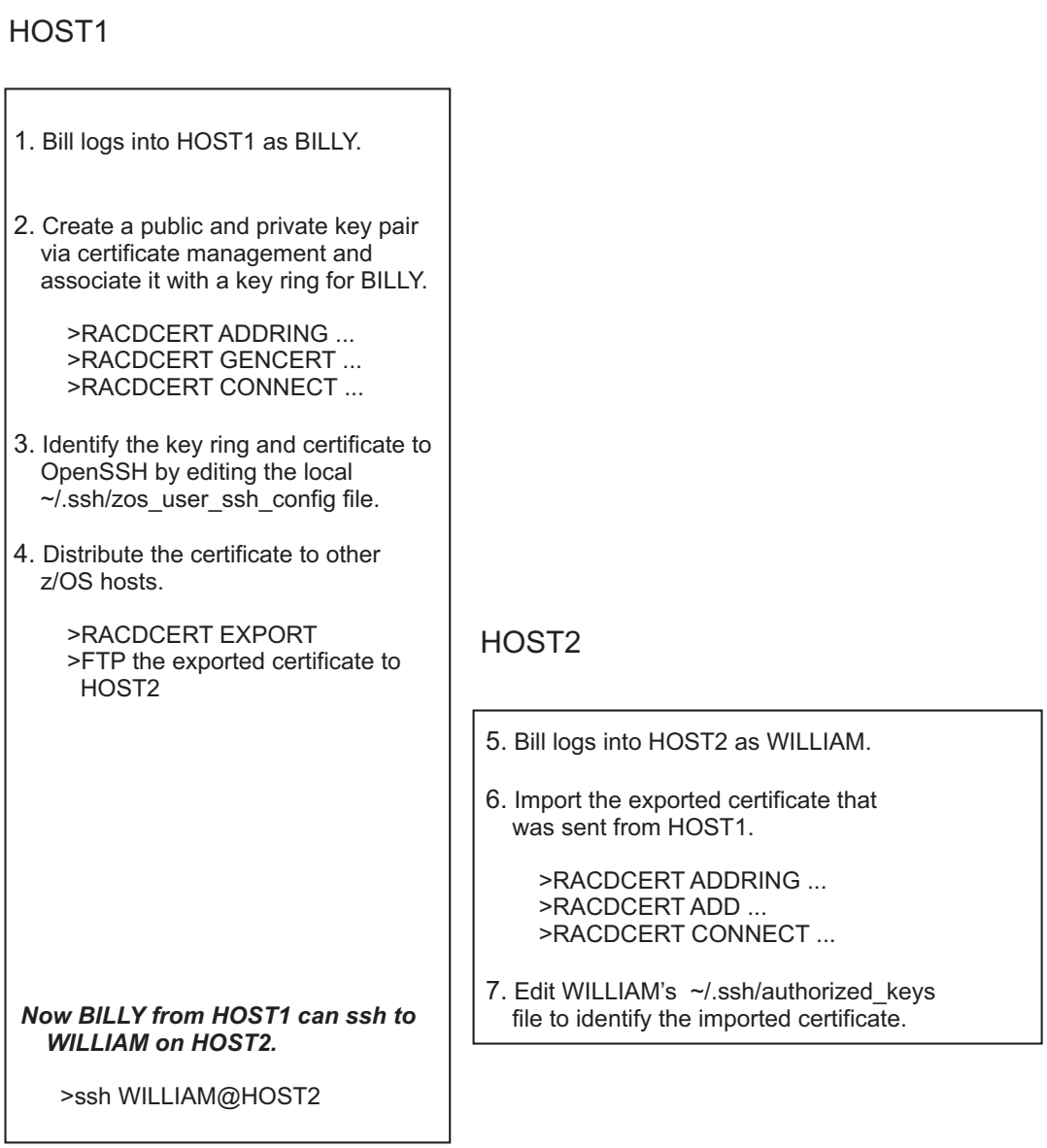

Figure 6. Accessing a remote system using ssh with public key authentication when keys are stored in real key rings

Steps for configuring your setup for X11 forwarding

X11 forwarding allows users who have an account on a UNIX machine to open a connection to the X11 interface remotely from another computer. Because this connection uses SSH, the communication between the systems is encrypted. X11 forwarding will only work if the system being connected to has both SSH and X11 forwarding enabled.

Before you begin: You need to know whether the system administrator has configured **sshd** on the remote host for X11 forwarding as described in "Steps for configuring the system for X11 forwarding" on page 47.

Perform the following steps to configure your system for X11 forwarding.

1. Enable X11 forwarding for your local SSH client. You can do this in one of two ways:
   a. Set the ForwardX11 configuration variable to yes in your `~/.ssh/config` file. This can be done on a per-host basis. This is useful if you want to always enable X11 forwarding.
   b. Invoke **ssh** with the **-X** option. Use this if you want to enable X11 forwarding for this session only.

---

2. In your local SSH configuration file (`~/.ssh/config`), specify the location of the xauth program on the remote system. This step is required only if the xauth program is installed somewhere other than the default location (`/usr/X11R6/bin/xauth`). The xauth program might need to support the generate command in order to allow **ssh** to successfully set up untrusted X11 forwarding, which is the default.

   Example: Following is an example of a **ssh** configuration file entry, using the default xauth location:

   ```
   XAuthLocation /usr/X11r6/bin/xauth
   ```

---

3. In your remote user account, if xauth is compiled to use DLLs, then set LIBPATH in `~/.ssh/environment` to include `/usr/lib`.

   Example:

   ```
   LIBPATH=/usr/lib
   ```

When you are done, you have configured your setup for X11 forwarding.

Chapter 9. OpenSSH command descriptions

scp — Secure copy (remote file copy program)

Format

scp [**–1246BCpqrv**] [**–c** *cipher*] [**–F** *ssh_config*] [**–i** *identity_file*] [**–l** *limit*] [**–o** *ssh_option*] [**–P** *port*] [**–S** *program*] [[*user@*]*host1*:]*file1 ...* [[*user@*]*host2*:]*file2*

Description

scp copies files between hosts on a network. It uses **ssh** for data transfer and uses the same authentication and provides the same security as **ssh**. **rcp** (remote copy) is a traditional UNIX utility that allows a user to copy files between remote hosts. Copies between two remote hosts are also permitted. When copying between two remote hosts, only options **-v**, **-r** and **-p** are passed to the remote host regardless of what the user specifies on the command line. Unlike **rcp**, **scp** asks for passwords, password phrases, or passphrases if they are needed for authentication.

File names can contain a user and host specification to indicate that the file is to be copied to the host or from the host. To prevent **scp** from treating the names containing ':' as specifiers, local file names can be made explicit by using absolute or relative path names.

IPv6 addresses can be specified by enclosing the address in square brackets.

scp assumes that files are text. Files copied between EBCDIC and ASCII platforms are converted.

If the source path name is a symbolic link, **scp** copies the file to which the symbolic link points. In other words, symbolic links are followed.

OpenSSH can be configured to collect SMF client and server transfer completion records that are associated with **scp**. See "Setting up OpenSSH to collect SMF records" on page 50 for more information. See Chapter 12, "SMF Type 119 records for OpenSSH," on page 167 for more information about the SMF client and server transfer completion records (subtypes 97 and 96 respectively). SMF records are not collected for local-to-local copies.

Restriction: The maximum full path name length is 1023 bytes for files processed by **scp**. Exceeding this maximum might result in unexpected behavior.

Options

–1 Specifies that **scp** is to use protocol version 1 only.

–2 Specifies that **scp** is to use protocol version 2 only.

–4 Forces **scp** to use IPv4 addresses only. If both **–4** and **–6** are specified, **scp** uses the option that appears last on the command line.

–6 Forces **scp** to use IPv6 addresses only. If both **–4** and **–6** are specified, **scp** uses the option that appears last on the command line.

–B Selects batch mode; while in batch mode, prompts are not issued for passwords, password phrases, or passphrases, but they are still required

for OpenSSH. To avoid password prompts, use public-key authentication with an **ssh-agent** or host-based authentication.

–c *cipher*
Selects the cipher to use for encrypting the data transfer. This option is directly passed to **ssh**. For more information, see the **ssh** "-c option" on page 87 or the **ssh_config** keyword "Ciphers" on page 130.

–C Enables compression. Passes the **–C** flag to **ssh** to enable compression.

–F *ssh_config*
Specifies an alternative per-user configuration file for **ssh**. This option is directly passed to **ssh**. This option has no effect on the z/OS-specific configuration files.

–i *identity_file*
Selects the file from which the identity (private key) for RSA or DSA authentication is read. This option is directly passed to **ssh**. For more information, see "ssh" on page 85.

–l Limits the used bandwidth, specified in Kbits.

–o *ssh_option*
Can be used to pass options to **ssh** in the format used in the **ssh_config** configuration file. This option is useful for specifying options for which there is no separate **scp** command-line flag. For full details of the available options and their values, see "ssh_config" on page 129. The z/OS-specific per-user OpenSSH client configuration options (see "zos_user_ssh_config" on page 142) can be specified on **-o**, but the z/OS-specific system-wide options (see "zos_ssh_config" on page 141) cannot.

Examples:

1. To use protocol version 1:
   ```
   scp -oProtocol=1
   ```
2. To disable password authentication:
   ```
   scp -oPasswordAuthentication=no
   ```

–p Preserves modification times, access times, and modes from the original file.

–P *port*
Specifies the port to connect to on the remote host.

–q Quiet. Disables the progress meter as well as the warning and diagnostic messages from **ssh**.

–r Recursively copies entire directories.

–S *program*
Name of program to use for the encrypted connection. The program must understand **ssh** options.

–v Verbose mode. Causes **scp** and **ssh** to print debugging messages about their progress, which is helpful in debugging connection, authentication, and configuration problems.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SMF_FD
Set to the file descriptor number used for interprocess communication during SMF-related processing. This environment variable is only used internally and is not for external specification.

Exit values

0 Successful completion

>0 An error occurred.

Related information

sftp, **ssh**, **sshd**, **ssh-add**, **ssh-agent**, **ssh_config**, **ssh-keygen**, **zos_ssh_config**, **zos_user_ssh_config**

Authors

Timo Rinne and Tatu Ylonen

---

sftp — Secure file transfer program

Format

sftp [**1Cv**] [**–B** *buffer_size*] [**–b** *batchfile*] [**–F** *ssh_config*] [**–o** *ssh_option*] [**–P** *sftp_server_path*] [**–R** *num_requests*] [**–S** *program*] [**–s** *subsystem | sftp_server*] *host*

sftp [[*user@*]*host*[:*file* [*file*]]]

sftp [[*user@*]*host*[:*dir*[/]]]

sftp -b batchfile [*user@*]*host*

Description

sftp is an interactive file transfer program similar to **ftp** which performs all operations over an encrypted **ssh** transport. It uses many features of **ssh**, such as public key authentication and compression.

sftp connects and logs into the specified host and then enters a subcommand mode.

- The second usage format retrieves files automatically if a non-interactive authentication method is used; otherwise it does so after successful interactive authentication.
- The third usage format allows **sftp** to start in a remote directory.
- The fourth usage format allows for automated sessions using the **-b** option. In such cases, you might have to configure public key authentication to eliminate the need to enter a password at connection time. For more information, see “sshd” on page 116 and “ssh-keygen” on page 105.

IPv6 addresses can be specified by enclosing the address in square brackets.

By default, **sftp** assumes files are binary. Files copied between EBCDIC and ASCII platforms are not converted. Use the ascii subcommand to transfer files in ASCII between the local host and the remote host.

OpenSSH can be configured to collect SMF client transfer completion records that are associated with **sftp**. See "Setting up OpenSSH to collect SMF records" on page 50 for more information. See Chapter 12, "SMF Type 119 records for OpenSSH," on page 167 for more information about the SMF client transfer completion records (subtype 97).

Restriction: The maximum full path name length is 1023 bytes for files processed by **sftp**. Exceeding this maximum might result in unexpected behavior.

Options

–1 Specifies the use of SSH protocol version 1. Because SSH protocol version 1 does not support subsystems, you must specify **–s** with an **sftp-server** path when using this option. This option is only supported if both the local and remote hosts are z/OS systems.

–b *batchfile*
Batch mode reads a series of commands from an input batchfile instead of stdin. Because it lacks user interaction, use it in conjunction with noninteractive authentication. A batchfile of '-' can be used to indicate standard input. **sftp** ends and the exit value is set to nonzero only if any of the following commands fail: **get**, **put**, **rename**, **ln**, **rm**, **rmdir**, **mkdir**, **cd**, **ls**, **lcd**, **chmod**, **chown**, **chgrp**, **lpwd** and **lmkdir**. For an exception, see "Limitations" on page 81.

Ending on error can be suppressed on a command-by-command basis by prefixing the command with a '-' character.

Example:

```
-rm /tmp/file*
```

–B *buffer_size*
Specifies the size of the buffer that **sftp** uses when transferring files. Larger buffers require fewer round trips at the cost of higher memory consumption. The default is 32768 bytes. If specifying buffer_size > INT_MAX, **sftp** only allocates INT_MAX at most. For more information, see "Limitations" on page 81.

–C Enables compression. This option is passed to **ssh**.

–F *ssh_config*
Specifies an alternative per-user **ssh_config** configuration file for **ssh**. This option is directly passed to **ssh**. It has no effect on the z/OS-specific configuration files.

–o *ssh_option*
Can be used to pass options to **ssh** in the format used in the **ssh_config** and **zos_user_ssh_config** configuration files. This is useful for specifying options for which there is no separate **sftp** command-line flag. For full details of the available options and their values, see "ssh_config" on page 129 and "zos_user_ssh_config" on page 142. The z/OS-specific per-user OpenSSH client configuration options can be specified on **-o**, but the z/OS-specific system-wide options (see "zos_ssh_config" on page 141) cannot.

Example: To specify an alternate port, use:

```
sftp -oPort=24
```

–P *sftp_server_path*
Connects directly to the local **sftp-server** (instead of via **ssh**). This option might be useful in debugging the client and server.

Restriction: When this option is specified, SMF client transfer completion records (subtype 97) are not collected.

–R *num_requests*
Specifies the number of requests that can be outstanding at any one time. Increasing this might slightly improve file transfer speed, but increases memory usage. The default is 16 outstanding requests.

–s *subsystem | sftp_server*
Specifies the SSH protocol version 2 subsystem or the path for an sftp server on the remote host. An **sftp-server** path is useful for using **sftp** over SSH protocol version 1 or when the remote **sshd** does not have an **sftp** subsystem configured.

–S *program*
Name of the program to use for the encrypted connection. The program must understand **ssh** options.

–v Enables verbose mode. This option is also passed to **ssh**. Multiple **–v** options increase the verbosity. You can specify up to three **-v** options.

Limitations

Due to limitations in the SECSH protocol with regards to EBCDIC platforms, **sftp** used with SSH protocol version 1 is only supported from z/OS to z/OS. (For information about the IETF SECSH internet drafts, see Appendix C, "RFCs and Internet drafts," on page 339).

The biggest buffer size that can be allocated is 2147483647(INT_MAX) bytes. INT_MAX is defined in limits.h.

When using **put -p** in conjunction with **-b**, if a failure occurs when preserving permissions or access time on the remote system, **sftp** will not exit and the exit value will not be set to nonzero.

Subcommands

sftp understands a set of commands (subcommands) similar to those of **ftp**.

Rules:

- Commands are not case sensitive.
- Path names that contain spaces must be enclosed in quotes.
- Glob characters (also called wildcard characters) in path names must be escaped with backslash characters (\). For more information about wildcard characters, refer to the section on file name generation in the **sh** command description in *z/OS UNIX System Services Command Reference*.

ascii Changes the data transfer type to ASCII.

For outgoing files, convert from EBCDIC code page of the current locale into ASCII before transferring them to the remote host. For incoming files, convert from ASCII into the code page of the current locale before restoring them on the local host.

Restriction: The ascii subcommand is only valid for file transfers between UNIX platforms. It is not valid for file transfers between Windows® and UNIX platforms.

binary Changes the data transfer type to binary. This is the default.

bye Quits **sftp**.

cd *path*
Changes the remote directory to *path*.

lcd *path*
Changes the local directory to *path*.

chgrp *grp path*
Changes group of file *path* to *grp*. *grp* must be a numeric GID. *path* can contain glob characters and match multiple files.

chmod *mode path*
Changes permissions of file *path* to *mode*. *path* can contain glob characters and match multiple files.

chown *own path*
Changes owner of file *path* to *own*. *own* must be a numeric UID. *path* can contain glob characters and match multiple files.

exit Quits **sftp**.

get [–Pp] *remote-path* **[***local-path***]**
Retrieves the *remote-path* and stores it on the local machine. If the local path name is not specified, it is given the same name it has on the remote machine. *remote-path* can contain glob characters and match multiple files. If it matches multiple files and *local-path* is specified, then *local-path* must specify a directory. If the **–P** or **–p** flag is specified, then the file's full permissions and access time are copied as well.

help Displays help text.

lls [*ls-options* **[***path***]]**
Displays local directory listing of either *path* or current directory if *path* is not specified. *ls-options* is case sensitive. *ls-options* can contain any flags supported by the local system's **ls** command. *path* can contain glob characters and match multiple files.

lmkdir *path*
Creates local directory specified by *path*.

ln *oldpath newpath*
Creates a symbolic link from *oldpath* to *newpath* on the remote host. Same as **symlink**.

lpwd Prints local working directory.

ls [–1aflnrSt] [*path***]**
Displays remote directory listing of either *path* or current directory if *path* is not specified. *path* can contain glob characters and match multiple files.

The following flags are recognized and the behavior of **ls** is altered accordingly:

-1 Produces single-column output.

-a Lists files beginning with a dot (.).

-f Does not sort the listing. The default sort order is lexicographical.

-l Displays additional details including permissions and ownership information.

-n Produces a long listing with user and group information presented numerically.

-r Reverses the sort order of the listing.

-S Sorts the listing by file size.

-t Sorts the listing by last modification time.

lumask *umask*
Sets local umask to *umask*.

mkdir *path*
Creates remote directory specified by *path*.

progress
Toggles display of progress meter.

put [–Pp] *local-path* **[***remote-path***]**
Uploads *local-path* and stores it on the remote machine. If the *remote-path* name is not specified, it is given the same name it has on the local machine. *local-path* can contain glob characters and match multiple files. If it matches multiple files and *remote-path* is specified, then *remote-path* must specify a directory. If the **–P** or **–p** flag is specified, then the file's permissions and access time are copied as well.

When using **put -p** with **-b**, if a failure occurs when preserving permissions or access time on the remote system, **sftp** will not exit and the exit value will not be set to nonzero.

pwd Displays the remote working directory.

quit Quits **sftp**.

rename *oldpath newpath*
Renames the remote file from *oldpath* to *newpath*.

rmdir *path*
Removes the remote directory specified by *path*.

rm *path*
Deletes the remote file specified by *path*.

symlink *oldpath newpath*
Creates a symbolic link from *oldpath* to *newpath* on the remote host. Same as **ln**.

version
Displays the **sftp** version.

! Escapes to local shell.

! *command*
Executes *command* in the local shell.

? Synonym for **help**.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SMF_FD
Set to the file descriptor number used for interprocess communication during SMF-related processing. This environment variable is only used internally and is not for external specification.

Exit values

0 Successful completion

>0 An error occurred. This exit value only occurs when **-b batchfile** is used and any of the following commands fail: **get**, **put**, **rename**, **ln**, **rm**, **rmdir**, **mkdir**, **cd**, **ls** ,**lcd**, **chmod**, **chown**, **chgrp**, **lpwd**, and **lmkdir**. For an exception, see "Limitations" on page 81.

Related information

scp, **ssh**, **ssh-add**, **ssh_config**, **ssh-keygen**, **sftp-server**, **sshd**, **zos_ssh_config**, **zos_user_ssh_config**

Author

Damien Miller

---

sftp-server — SFTP server subsystem

Format

sftp-server [**-eh**] [**-f** *log_facility*] [**-l** *log_level*]

Description

sftp-server is a program that implements the server side of the SFTP protocol. It expects client requests from standard input and writes responses to standard output. **sftp-server** is not intended to be called directly, but by specifying the **sshd_config** keyword Subsystem. See "Subsystem" on page 156 for more information about the keyword.

OpenSSH can be configured to collect SMF server transfer completion records that are associated with **sftp-server**. See "Setting up OpenSSH to collect SMF records" on page 50 for more information. See Chapter 12, "SMF Type 119 records for OpenSSH," on page 167 for more information about the SMF server transfer completion records (subtype 96).

Restriction: The maximum full path name length is 1023 bytes for files processed by **sftp-server**. Exceeding this maximum might result in unexpected behavior.

Options

–e **sftp-server** sends log messages to standard error instead of the system log.

–f *log_facility*
Specifies the facility code that is used when logging messages from **sftp-server**. The possible values are: DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The default is AUTH.

For more information about these log facilities, see the syslog daemon section in *z/OS Communications Server: IP Configuration Reference*.

–h Displays a summary of options.

–l *log_level*
Specifies which messages will be logged by **sftp-server**. The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. INFO and VERBOSE log transactions that **sftp-server** performs on behalf of the client. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of debugging output. The default is ERROR.

These logging levels are similar to the syslog daemon priority codes, which are described in the syslog daemon section in *z/OS Communications Server: IP Configuration Reference*.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SMF_FD
Set to the file descriptor number used for interprocess communication during SMF-related processing. This environment variable is only used internally and is not for external specification.

Related information

sftp, **ssh**, **sshd**, **sshd_config**, **zos_sshd_config**

Author

Markus Friedl

---

ssh — OpenSSH client (remote login program)

Format

ssh [**–1246AaCfgKkMNnqsTtVvXxY**] [**–b** *bind_address*] [**–c** *cipher_spec*] [**–D** [*bind-address:*]*port*] [**–e** *escape_char*] [**–F** *configfile*] [**–i** *identity_file*] [**–L** [*bind-address:*]*port:host:hostport*] [**–l** *login_name*] [**–m** *mac_spec*] [**-O** *ctl_cmd*] [**–o** *option*] [**–p** *port*] [**–R** [*bind-address:*] *port:host:hostport*] [**-S** *ctl_path*] [**-w** *local_tun* [*:remote_tun*]] [*user@*] *hostname* [*command*]

Description

ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine. It is an alternative to **rlogin** and **rsh** and provides secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified host name (with optional user name). If *command* is specified, instead of a login shell being executed, *command* is executed

on the remote host. Users must prove their identity to the remote machine using one of several methods, depending on the protocol version used.

Tip: To avoid problems when running as a user that shares a UID, run **ssh** with the **-F** option to specify a user-specific **ssh_config** file. The file should set the IdentityFile, User, and UserKnownHostsFile keywords to the proper user-specific values. You should also specify a user-specific **zos_user_ssh_config** file using the _ZOS_USER_SSH_CONFIG environment variable.

Options

–1 Forces **ssh** to try protocol version 1 only. If both **–1** and **–2** are specified, **ssh** uses the option that appears last on the command line.

–2 Forces **ssh** to try protocol version 2 only. If both **–1** and **–2** are specified, **ssh** uses the option that appears last on the command line.

–4 Forces **ssh** to use IPv4 addresses only. If both **–4** and **–6** are specified, **ssh** uses the option that appears last on the command line.

–6 Forces **ssh** to use IPv6 addresses only. If both **–4** and **–6** are specified, **ssh** uses the option that appears last on the command line.

–a Disables forwarding of the authentication agent connection.

–A Enables forwarding of the authentication agent connection. This can also be specified on a per-host basis in a **ssh_config** configuration file.

Guideline: Enable agent forwarding with caution. Users with the ability to bypass file permissions on the remote host (for the agent's UNIX-domain socket) can access the local agent through the forwarded connection. Attackers cannot obtain key material from the agent. However, they can perform operations on the keys that enable them to authenticate using the identities loaded into the agent.

–b *bind_address*
Use *bind_address* on the local machine as the source address of the connection. This option is useful only on systems with more than one address.

Rule: The *bind_address* must be the same address family (IPv4 or IPv6) as the remote host name specified on the **ssh** command line.

–c *cipher_spec*
Selects the cipher to use for encrypting the session.

For protocol 1 specifications:

3des 3des (Triple-DES) is an encrypt-decrypt-encrypt triple with three different keys. It is the default.

blowfish Blowfish is a secure fast block cipher.

des Specifying des is strongly discouraged due to cryptographic weakness. It is supported only in **ssh** for interoperability with legacy protocol 1 implementations that do not support the 3DES cipher.

For protocol version 2 specifications, ciphers can be specified in order of preference in a comma-separated list. Valid ciphers include:

3des-cbc Triple-DES (3DES) algorithm

acss@openssh.org
OpenSSH acss@openssh.org cipher

aes128-cbc Advanced Encryption Standard (AES) CBC mode with 128-bit key

aes128-ctr Advanced Encryption Standard (AES) CTR mode with 128-bit key

aes192-cbc Advanced Encryption Standard (AES) CBC mode with 192-bit key

aes192-ctr Advanced Encryption Standard (AES) CTR mode with 192-bit key

aes256-cbc Advanced Encryption Standard (AES) CBC mode with 256-bit key

aes256-ctr Advanced Encryption Standard (AES) CTR mode with 256-bit key

arcfour Arcfour algorithm

arcfour128 Arcfour algorithm with 128-bit key

arcfour256 Arcfour algorithm with 256-bit key

blowfish-cbc Blowfish algorithm

cast128-cbc CAST algorithm

rijndael-cbc@lysator.liu.se
Same as Advanced Encryption Standard (AES) CBC mode with 256-bit key

The cipher is typically one long unbroken line; in the following example the cipher is not shown as one unbroken line due to space limitations. the default is:

```
aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,
3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,
rijndael-cbc@lysator.liu.se
```

–C Requests compression of all data (including stdin, stdout, stderr, and data for forwarded X11 and TCP connections). The compression level can be controlled by the CompressionLevel option for protocol version 1. The default value can be set on a per-host basis in the **ssh_config** configuration file; for more information about the Compression and CompressionLevel options, see “ssh_config” on page 129.

–D [*bind_address:***]***port*
Specifies a local dynamic application-level port forwarding. This type of dynamic port forwarding works by allocating a socket to listen to port on the local side, optionally bound to the specified *bind_address*. Whenever a connection is made to this port, it is forwarded over the secure channel and the application protocol is used to determine where to connect from the remote machine. Currently, the SOCKS4 and SOCKS5 protocol are supported and **ssh** will act as a SOCKS server. Only a superuser can forward privileged ports. Dynamic port forwardings can also be specified in the **ssh_config** configuration file.

IPv6 addresses can be specified with an alternative syntax: [*bind_address*/]*port* or by enclosing the address in square brackets. Only the superuser can forward privileged ports. By default, the local port is bound in accordance with the GatewayPorts setting. However, an explicit

bind_address can be used to bind the connection to a specific address. The *bind_address* of "localhost" indicates that the listening port is to be bound for local use only, while an empty address or '*' indicates that the port should be available from all interfaces.

Appendix B, "OpenSSH - port forwarding examples," on page 335 has examples of port forwarding.

–e *escape_char*
Sets the escape character for sessions with a pty (the default is " ~"). The escape character is only recognized at the beginning of a line. The escape character followed by a dot ('.') closes the connection, followed by Control-Z suspends the connection, and followed by itself sends the escape character once. Setting the character to "none" disables any escape characters and makes the session fully transparent.

–f Requests **ssh** to go to the background before command execution. This is useful if **ssh** is going to ask for passwords, password phrases, or passphrases, but the user wants it in the background. This implies **–n**. The recommended way to start X11 programs at a remote site is **ssh –f** *host xterm*.

–F *configfile*
Specifies an alternative per-user **ssh_config** configuration file. If an **ssh_config** configuration file is given on the command line, the system-wide **ssh_config** configuration file (`/etc/ssh/ssh_config`) will be ignored. The default for the per-user **ssh_config** configuration file is `~/.ssh/config`. This option has no effect on the z/OS-specific configuration files.

–g Allows remote hosts to connect to local forwarded ports.

–i *identity_file*
Selects a file from which the identity (private key) for RSA or DSA authentication is read. The default is `~/.ssh/identity` for protocol version 1. For protocol version 2 , the default is `~/.ssh/id_rsa` and `~/.ssh/id_dsa`. Identity files can also be specified on a per-host basis in the **ssh_config** configuration file. It is possible to have multiple **–i** options (and multiple identities specified in the **ssh_config** configuration file).

For a given protocol, identity files are tried in the order they are specified. If key ring certificates have been separately specified, then they will always be tried before identity files. The certificates are used in the order they were specified, followed by the identity files in the order they were specified. The key ring certificates could be specified either via a command-line option by specifying one or more IdentityKeyRingLabel options on the **-o** option, or by specifying the IdentityKeyRingLabel keyword in the **zos_user_ssh_config** file (the z/OS-specific per-user client configuration file).

However, if an identity is loaded in an agent, regardless of whether it originated from a key ring certificate or from a file, then that identity will be tried first.

To sum it up, the order that identities are tried are as follows:

1. Identities in the agent.
2. The key ring certificates on the command-line option
3. Key ring certificates specified in a **zos_user_ssh_config** file
4. Identity files on the command-line option, and then

5. Identity files specified in an **ssh_config** configuration file.

–I *smartcard_device*
(**–I** is the uppercase **– i**). Not supported on z/OS UNIX. Specifies which smart card device to use. The argument is the device that **ssh** should use to communicate with a smart card used for storing the user's private RSA key.

-k Not supported on z/OS UNIX. Disables forwarding (delegation) of GSS-API credentials to the server.

GSS-API stands for Generic Security Services Application Programming Interface. It is a generic API for handling client-server authentication. Because it provides security services to callers in a generic way, supportable with a range of underlying mechanisms and technologies, it allows for source-level portability of applications to different environments. For more details, check IETF standard RFC 2743 at http://www.ietf.org/rfc/rfc2743.txt.

-K Not supported on z/OS UNIX. Enables GSS-API authentication and forwarding (delegation) of GSS-API credentials to the server

–l *login_name*
Specifies the user to log in as on the remote machine. This option can also be specified on a per-host basis in the **ssh_config** configuration file.

–L [*bind-address:***]***port***:***host***:***hostport*
Specifies that *port* on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to *port* on the local side, optionally bound to the specified *bind_address*. When a connection is made to this port, it is forwarded over the secure channel and a connection is made to *host port hostport* from the remote machine. Port forwardings can also be specified in the **ssh_config** configuration file. Only a superuser can forward privileged ports.

IPv6 addresses can be specified with an alternative syntax: [*bind_address/*]*port/host/hostport* or by enclosing the address in square brackets.

By default, the local port is bound in accordance with the GatewayPorts setting. However, an explicit *bind_address* can be used to bind the connection to a specific address. The *bind_address* of "localhost" indicates that the listening port be bound for local use only, while an empty address or '*' indicates that the port should be available from all interfaces.

Appendix B, “OpenSSH - port forwarding examples,” on page 335 has examples of port forwarding.

–m *mac_spec*
For protocol version 2, a comma-separated list of MAC (message authentication code) algorithms can be specified in order of preference. **ssh_config** contains a description of “MACs” on page 136.

-M Places the **ssh** client into master mode for connection sharing. Multiple **-M** options puts **ssh** into master mode with confirmation required before slave connections are accepted. **ssh_config** contains a description of “ControlMaster” on page 131.

–n Redirects stdin from `/dev/null` (prevents reading stdin). This option must be used when **ssh** is run in the background. A common trick is to use this to run X11 programs on a remote machine.

Example:

```
ssh –n shadows.cs.hut.fi emacs &
```

Result: An emacs session is started on shadows.cs.hut.fi and the X11 connection is automatically forwarded over an encrypted channel. The **ssh** program is put in the background. This does not work if **ssh** needs to ask for a password, password phrase, or passphrase; see the **–f** option.

–N Specifies that a remote command not be executed. This is useful for just forwarding ports (protocol version 2 only). This option overrides the **-t** option.

–o *option*
Can be used to give options in the format used in the **ssh_config** and **zos_user_ssh_config** configuration files. This is useful for specifying options for which there is no separate command-line flag. For full details of the available options and their values, see “ssh_config” on page 129 and “zos_user_ssh_config” on page 142. The z/OS-specific per-user OpenSSH client configuration options can be specified on **-o**, but the z/OS specific system-wide options (see “zos_ssh_config” on page 141) cannot.

Example:

```
ssh -oHostbasedAuthentication=no Billy@us.pok.ibm.com
```

-O *ctl_cmd*
Controls the master process of a multiplexed connection. When the **-O** option is specified, the *ctl_cmd* argument is interpreted and passed to the master process. Valid commands are "check" (check that the master process is running) and "exit" (request the master to exit).

–p *port*
Port to connect to on the remote host. This can be specified on a per-host basis in the **ssh_config** configuration file.

–q Quiet mode. Suppresses most warning and diagnostic messages.

–R [*bind_address:***]***port***:***host***:***hostport*
Specifies that the given port on the remote (server) host is to be forwarded to host and port on the local side. A socket is allocated to listen to *port* on the remote side; when a connection is made, it is forwarded over the secure channel and a connection is made to *host port hostport* from the local machine. Port forwardings can also be specified in the **ssh_config** configuration file. Privileged ports can be forwarded only when logging in as superuser on the remote machine. IPv6 addresses can be specified by enclosing the address in square brackets or using an alternative syntax: [*bind_address/*]*port/host/hostport*.

By default, the listening socket on the server is bound to the loopback interface only. The default can be overridden by specifying a *bind_address*. An empty *bind_address*, or the address '*', indicates that the remote socket should listen on all interfaces. Specifying a remote *bind_address* will only succeed if the server's GatewayPorts option is enabled as described in “GatewayPorts” on page 133.

–s Can be used to request invocation of a subsystem on the remote system. Subsystems are a feature of SSH protocol version 2, which facilitates the use of **ssh** as a secure transport for other applications such as **sftp**. The subsystem is specified as the remote command.

Example:

```
ssh -s host subsystem_name
```

User-defined subsystems (those that are not built-in) are only supported when both the OpenSSH client and server are running on a z/OS system. See “Limitations” on page 95 for more information.

-S *ctl_path*
Specifies the location of a control socket for connection sharing. For more information, see the descriptions of the **ssh_config** keywords “ControlMaster” on page 131 and “ControlPath” on page 132.

–t Forces pty allocation. This option can be used to execute arbitrary screen-based programs on a remote program, which can be very useful, for example, when implementing menu services. Multiple **–t** options force pty allocation, even if **ssh** has no local tty. Both single and multiple uses of **–t** will be overridden by either the **–T** or **–N** options.

–T Disables pty allocation. This option overrides the **–t** option.

–v Verbose mode. Causes **ssh** to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems. Multiple **–v** options increase the verbosity. You can specify up to three **-v** options.

–V Displays the current OpenSSH and OpenSSL version information and exits.

-w *local_tun*[:*remote_tun*]
Not supported on z/OS UNIX. Requests tunnel device forwarding with the specified devices between the client (*local_tun*) and the server (*remote_tun*).

The devices can be specified by numerical ID or the keyword "any", which uses the next available tunnel device. If *remote_tun* is not specified, it defaults to "any". See also the descriptions of the **ssh_config** options “Tunnel” on page 139 and “TunnelDevice” on page 140. If the Tunnel option is unset, it is set to the default tunnel mode, which is "point-to-point".

–x Disables X11 forwarding.

–X Enables X11 forwarding. This can also be specified on a per-host basis in the **ssh_config** configuration file.

X11 forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the user's X authorization database) can access the local X11 display through the forwarded connection. An attacker may then be able to perform activities such as keystroke monitoring.

For this reason, X11 forwarding is subjected to X11 SECURITY extension restrictions by default. See the description of the **ssh -Y** option and the **ssh_config** option “ForwardX11Trusted” on page 133 for more information.

–Y Enables trusted X11 forwarding. Trusted X11 forwardings are not subjected to the X11 SECURITY extension controls.

ssh can additionally obtain **ssh_config** configuration data from a per-user configuration file and a system-wide **ssh_config** configuration file. For file format and configuration options, see “ssh_config” on page 129. **ssh** can also obtain z/OS-specific configuration data from a system-wide **zos_ssh_config** configuration file and per-user **zos_user_ssh_config** configuration file. For file format and configuration options, see “zos_ssh_config” on page 141 and “zos_user_ssh_config” on page 142.

Host key checking

In host key checking, **ssh** automatically maintains and checks a database containing identification for all hosts it has ever been used with. Host keys are stored in `~/.ssh/known_hosts` in the user's home directory. Additionally, the `/etc/ssh/ssh_known_hosts` file is automatically checked for known hosts. Any new hosts can be automatically added to the user's file. If a host's identification changes, **ssh** warns about this and disables password authentication to prevent server spoofing or man-in-the-middle attacks, which could otherwise be used to circumvent the encryption. The **ssh_config** keyword StrictHostKeyChecking can be used to control logins to machines whose host key is not known or has changed. The keyword is described in "StrictHostKeyChecking" on page 139.

Authentication

The OpenSSH SSH client supports SSH protocol version 1 and protocol version 2. Protocol version 2 is the default. These settings can be altered using the **ssh_config** Protocol option (described in "Protocol" on page 137), or enforced using the -1 and -2 options. Both protocols support similar authentication methods, but protocol version 2 is preferred because it provides additional mechanisms for confidentiality (the traffic is encrypted using, for example, AES, 3DES, Blowfish, CAST128, or Arcfour) and integrity (for example, hmac-md5, hmac-sha1, umac-64, hmac-ripemd160). Protocol version 1 lacks a strong mechanism for ensuring the integrity of the connection.

The methods available for authentication are:

- Host-based authentication (disabled by default). See "Host-based authentication."
- Public key authentication. See "Public key authentication" on page 93.
- Challenge-response authentication (not supported on z/OS UNIX). See "Challenge-response authentication" on page 93.
- Password authentication. See "Password authentication" on page 93.

Authentication methods are tried in the order listed above, though protocol version 2 has a configuration option to change the default order: the **sshd_config** keyword PreferredAuthentications. The keyword is described in "PreferredAuthentications" on page 137.

Host-based authentication

In host-based authentication, if the machine the user logs in from is listed in `/etc/hosts.equiv` or `/etc/shosts.equiv` on the remote machine, and the user names are the same on both sides, or if the files `~/.rhosts` or `~/.shosts` exist in the user's home directory on the remote machine and contain a line containing the name of the client machine and the name of the user on that machine, the user is considered for login. Additionally, the server must be able to verify the client's host key for the login to be permitted. (See the description of "~/.ssh/known_hosts" on page 97 and "/etc/ssh/ssh_known_hosts" on page 97.) This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing.

For more information about host-based authentication, refer to the **ssh_config** keyword "HostbasedAuthentication" on page 134.

Guideline: The `/etc/hosts.equiv`, `~/.rhosts`, and **rlogin**/**rsh** protocol in general, are inherently insecure and the administrator should disable them if security is desired.

Public key authentication

In public key authentication, the scheme is based on public key cryptography, using cryptosystems where encryption and decryption are done using separate keys, and it is not feasible to derive the decryption key from the encryption key. Each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key. **ssh** implements public key authentication protocol automatically, using either the RSA or DSA algorithms. Protocol version 1 is restricted to using only RSA keys, but protocol version 2 can use either.

The `~/.ssh/authorized_keys` file lists the public keys that are permitted for logging in. When the user logs in, **ssh** tells the server which key pair it would like to use for authentication. The client proves that it has access to the private key and the server checks that the corresponding public key is authorized to accept the account.

One method of creating a key pair is by running **ssh-keygen**. This action stores the private key in `~/.ssh/identity` (protocol version 1), `~/.ssh/id_dsa` (protocol version 2 DSA), or `~/.ssh/id_rsa` (protocol version 2 RSA) and stores the public key in `~/.ssh/identity.pub` (protocol version 1), `~/.ssh/id_dsa.pub` protocol version 2 DSA), or `~/.ssh/id_rsa.pub` (protocol version 2 RSA) in the user's home directory. The user then copies the public key to the `~/.ssh/authorized_keys` file in the home directory on the remote machine. The `authorized_keys` file corresponds to the conventional `~/.rhosts` file, and has one key per line, though the lines can be very long. After this, the user can log in without giving the password.

Another method of creating a key pair is by using digital certificates associated with a SAF key ring, either real or virtual. See “Steps for setting up user authentication when keys are stored in key rings” on page 68 for more information about using SAF key rings to manage your keys.

The most convenient way to use public key authentication might be with an authentication agent. See “ssh-agent” on page 102 for more information.

Challenge-response authentication

In challenge-response authentication, the server sends an arbitrary challenge text and prompts for a response. Protocol version 2 allows multiple challenges and responses; protocol version 1 is restricted to just one challenge and response. Examples of challenge-response authentication include BSD Authentication and PAM (on some non-OpenBSD systems).

Challenge-response authentication is not supported on z/OS UNIX.

Password authentication

Finally, if other authentication methods fail, **ssh** prompts the user for a password and password phrase. The password and password phrase are sent to the remote host for checking; however, because all communications are encrypted, the password and password phrase cannot be seen by anyone listening on the network.

Login session and remote execution

When the user's identity has been accepted by the server, the server either executes the given command or logs into the machine and gives the user a normal shell on the remote machine. All communication with the remote command or shell is automatically encrypted.

If a pseudo terminal (pty) has been allocated (normal login session), the user can use the escape characters in “Escape characters.”

If no pty has been allocated, the session is transparent (escape characters are not recognized) and can be used to reliably transfer binary data. Setting the escape character to “none” will also make the session transparent even if a tty is used.

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 **ssh**.

Escape characters

When a pty has been requested, **ssh** supports a number of functions through the use of an escape character.

A single tilde character can be sent as "~~" or by following the tilde by a character other than those described below. The escape character must always follow a newline to be interpreted as a special character. The escape character can be changed in configuration files using the EscapeChar configuration option or on the command line by the **–e** option.

The supported escape characters (assuming the default "~") are:

~. Disconnect.

~^Z Background **ssh**.

~& Background **ssh** at logout when waiting for forwarded connections or X11 sessions to terminate.

~# List forwarded connections.

~? Display a list of escape characters.

~B Send a BREAK to the remote system.

Restriction: The ~B escape character is useful only for protocol version 2 and if the peer supports it.

~C Open command line. Use this option to do the following:

- Add port forwardings using the **-L** and **-R** options (see “-L option” on page 89 and “-R option” on page 90).
- Cancel existing remote forwardings using the **-KR** option (for example, **-KR***[bind_address:]port*).
- Execute a local command if the **ssh_config** keyword PermitLocalCommand enables the feature (for example, *!command*).
- Get basic help using the **-h** option.

~R Request rekeying of the connection.

Restriction: The ~R escape character is useful only for protocol version 2 and if the peer supports it.

X11 forwarding

If the ForwardX11 keyword is set to "yes" (or, see the description of the **-X**, **-x**, and **-Y** options described in “Options” on page 86) and X11 is in use (the DISPLAY environment variable is set), then the connection to the X11 display is automatically forwarded to the remote side. As a result, any X11 program that is started from the shell (or command) goes through the encrypted channel and the

connection to the real X server is made from the local machine. The user should not manually set DISPLAY. Forwarding of X11 connections can be configured on the command line or in configuration files. For more information about OpenSSH client configuration files, see “ssh_config” on page 129.

The DISPLAY value set by **ssh** points to the server machine, but with a display number greater than zero. This is normal and happens because **ssh** creates a proxy X server on the server machine for forwarding the connections over the encrypted channel. In other words, the **ssh** server masquerades as an X server.

ssh also automatically sets up Xauthority data on the server machine. For this purpose, it generates a random authorization cookie, stores it in Xauthority on the server, and verifies that any forwarded connections carry this cookie and replace it with the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent without encryption).

If the ForwardAgent variable is set to "yes" (or, see the description of the **-A** and **-a** options) and the user is using an authentication agent, the connection to the agent is automatically forwarded to the remote side.

TCP forwarding

Forwarding of arbitrary TCP connections over the secure channel can be specified either on the command line or in a configuration file. One possible application of TCP forwarding is a secure connection to a mail server; another is going through firewalls. For more information, see Appendix B, “OpenSSH - port forwarding examples,” on page 335.

Running OpenSSH in other locales

Rule: All files used by OpenSSH (such as key files and configuration files) must be in the IBM-1047 code set, with the exception of the **rc** files (`/etc/ssh/sshrc` and `~/.ssh/rc`). The **rc** files are parsed by /bin/sh and should be in the code set of the current locale. Do not use the `/etc/ssh/sshrc` file if there is a possibility of the users on the system running in different locales.

Limitations

User-defined subsystems are only supported when both the OpenSSH client and server are running on z/OS. This is due to a limitation in the SECSH protocol with regards to EBCDIC platforms; for more information about the IETF SECSH RFCs and internet drafts, see Appendix C, “RFCs and Internet drafts,” on page 339. User-defined subsystems are specified by using the **sshd_config** Subsystem keyword. Only the built-in **sftp** subsystem is supported for transfers between all platforms.

Restrictions:

- OpenSSH does not run in multibyte locales.
- The SSH client cannot be run from OMVS (which is a 3270 session). **ssh** has been disabled under OMVS because in some situations, passwords are visible while they are being typed by the user.

Examples

When passing shell commands on the SSH invocation line, the backslash escape character is needed to handle the characteristics of specifying a sequential data set or member of a partitioned data set (PDS).

- Copying from the z/OS UNIX file system to a PDS:

```
ssh user@ibm.com "cp ssh.log \"//'USER.SSH.LOG(LOG1)'\" "
```

- Copying from the z/OS UNIX file system to a sequential data set:

```
ssh user@ibm.com "cp ssh.log \"//'USER.SSH.LOG2'\" "
```

Files

~/.rhosts

This file is used for host-based authentication. On some machines, this file may need to be world-readable if the user's home directory is on an NFS partition, because **sshd** reads it as a superuser. Additionally, this file must be owned by the user and must not have write permissions for anyone else. The recommended permission for most machines is read/write for the user and not accessible by others.

~/.shosts

This file is used in exactly the same way as `~/.rhosts`, but allows host-based authentication without permitting login with **rlogin** or **rsh**.

~/.ssh/ This directory is the default location for all user-specific configuration and authentication information. There is no general requirement to keep the entire contents of this directory secret, but the recommended permissions are read/write/execute for the user, and not accessible by others.

~/.ssh/authorized_keys

Lists the public keys (RSA/DSA) that can be used for logging in as this user. For the format of this file, see “Format of the authorized_keys file” on page 120. The content of this file is not highly sensitive, but the recommended permissions are read/write for the user, and not accessible by others.

If this file, the `~/.ssh/` directory, or the user's home directory are writable by other users, then the file could be modified or replaced by unauthorized users. In this case, **sshd** will not allow it to be used unless the value for the **sshd_config** keyword StrictModes has been set to "no".

~/.ssh/config

The per-user **ssh_config** configuration file. The file format and configuration options are described in “ssh_config” on page 129. Because of the potential for abuse, this file must have strict permissions: read/write for the user, and not writable by others.

~/.ssh/environment

Contains additional definitions for environment variables. For more information, see “Environment variables” on page 98.

~/.ssh/identity, ~/.ssh/id_dsa, ~/.ssh/id_rsa

Contains the private key for authentication. These files contain sensitive data and should be readable by the user but not accessible by others (read/write/execute). Note that **ssh** ignores a private key file if it is accessible by others. It is possible to specify a passphrase when generating the key; the passphrase will be used to encrypt the sensitive part of this file using 3DES.

~/.ssh/identity.pub, ~/.ssh/id_dsa.pub, ~/.ssh/id_rsa.pub

Contains the public key for authentication. These files are not sensitive and can (but need not) be readable by anyone. The contents of the `~/.ssh/identity.pub` file must be added to `~/.ssh/authorized_keys` on all machines where the user wants to log in using protocol RSA

authentication. The contents of the ~/.ssh/id_dsa.pub and ~/.ssh/id_rsa.pub file must be added to ~/.ssh/authorized_keys on all machines where the user wants to log in using protocol version 2 DSA/RSA authentication. These files are never used automatically and are not necessary; they are only provided for the convenience of the user.

~/.ssh/known_hosts
Contains a list of host keys for all hosts that the user has logged into that are not already in the system-wide list of known host keys, /etc/ssh/ssh_known_hosts, which is described in "ssh_known_hosts file format" on page 122. This file should be writable only by the owner and the owner must be the user. It can be, but need not be, world-readable.

~/.ssh/rc
Commands in this file are executed by **ssh** when the user logs in, just before the user's shell (or command) is started. For more information about the format, see "Files" on page 124.

~/.ssh/zos_user_ssh_config
The z/OS-specific per-user client configuration file. The file format and configuration options are described in "zos_user_ssh_config" on page 142. Because of the potential for abuse, this file must have strict permissions: read/write for the user, and not writable by others.

/etc/hosts.equiv
This file is for host-based authentication. It should only be writable by a superuser. For more information about the format, see "Files" on page 124.

/etc/ssh/shosts.equiv
This file is used in exactly the same way as /etc/hosts.equiv but allows host-based authentication without permitting login with **rlogin** or **rsh**.

/etc/ssh/ssh_config
System-wide **ssh_config** configuration file. For file format and configuration information, see "ssh_config" on page 129.

/etc/ssh/ssh_host_key, /etc/ssh/ssh_host_dsa_key, /etc/ssh/ssh_host_rsa_key
These three files contain the private parts of the host keys and are used for host-based authentication. If protocol version 1 is used, **ssh** must be setuid 0 because the host key is readable only by a superuser. For protocol version 2, **ssh** uses **ssh_keysign** to access the host keys. This eliminates the requirement that **ssh** be setuid 0 when the host-based authentication is used. By default, **ssh** is not setuid 0.

/etc/ssh/ssh_known_hosts
System-wide list of known host keys. This file must be prepared by the system administrator to contain the public host keys of all machines in the organization, and it must be world-readable. For more information about the format, see "ssh_known_hosts file format" on page 122.

The canonical system name (as returned by name servers) is used by **sshd** to verify the client host when logging in; other names are needed because **ssh** does not convert the user-supplied name to a canonical name before checking the key, because someone with access to the name servers would then be able to fool host authentication.

/etc/ssh/sshrc
Commands in this file are executed by **ssh** when the user logs in, just before the user's shell (or command) is started. For more information about the format, see "Files" on page 124.

/etc/ssh/zos_ssh_config
z/OS-specific system-wide client configuration file. For file format and configuration information, see “zos_ssh_config” on page 141.

Environment variables

ssh typically sets or uses the following environment variables:

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SMF_FD
Set to the file descriptor number used for interprocess communication during SMF-related processing. This environment variable is only used internally and is not for external specification.

_ZOS_USER_SSH_CONFIG
Specifies the path name of the z/OS-specific per-user OpenSSH client configuration file. The system-wide default is `/etc/ssh/zos_ssh_config` and the user's default is `~/.ssh/zos_user_ssh_config`. If this variable is specified, it replaces the user's default file but not the system-wide default file. See “zos_ssh_config” on page 141 and “zos_user_ssh_config” on page 142 for the available keywords. The recommended permissions of the specified file are read/write for the user and not writable by others.

DISPLAY
Indicates the location of the X11 server. It is automatically set by **ssh** to point to a value of the form *hostname*:*n* where *hostname* indicates the host where the shell runs, and *n* is an integer greater than or equal to 1. **ssh** uses this special value to forward X11 connections over the secure channel. The user should normally not set DISPLAY explicitly, as that will render the X11 connection insecure (and require the user to manually copy any required authorization cookies).

HOME
Set to the path for the user's home directory.

LOGNAME
Synonym for USER.

MAIL Set to the path of the user's mailbox.

PATH Set to the default PATH, as compiled into **ssh**.

SSH_ASKPASS
If **ssh** needs a passphrase, it reads the passphrase from the current terminal if it was run from a terminal. If **ssh** does not have a terminal associated with it, but DISPLAY and SSH_ASKPASS are set, it executes the program specified by SSH_ASKPASS and opens an X11 window to read the passphrase. This is particularly useful when calling **ssh** from an .Xsession or related script. It is necessary to redirect the input from `/dev/null` to make this work.

SSH_AUTH_SOCK
Identifies the path of a UNIX-domain socket used to communicate with the agent.

SSH_CONNECTION
Identifies the client and server ends of the connection. The variable contains four space-separated values: client ip-address, client port number, server ip-address and server port number.

SSH_ORIGINAL_COMMAND
Contains the original command line if a forced command is executed. It can be used to extract the original arguments.

SSH_TTY
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 Set to indicate the present time zone if it was set when the daemon was started (the daemon passes the value on to new connections).

USER Set to the name of the user logging in.

Additionally, **ssh** reads `~/.ssh/environment` and adds lines of the format **VARNAME=***value* to the environment if the file exists and if users are allowed to change their environment. For more information, see "PermitUserEnvironment" on page 154.

Exit values

ssh exits with the exit status of the remote command or with 255 if an error occurred.

Related information

scp, **sftp**, **ssh-add**, **ssh-agent**, **ssh_config**, **ssh-keygen**, **ssh-keysign**, **sshd**, **zos_ssh_config**, **zos_user_ssh_config**

Authors

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

ssh-add — Add RSA or DSA identities to the authentication agent

Format

ssh-add [**–cDdLlXx**] [**–t** *life*] [*file ...*]
ssh-add –s *reader*
ssh-add –e *reader*

Description

ssh-add adds RSA or DSA identities to the authentication agent, **ssh-agent**. When run without arguments and when neither of the key ring environment variables is set, it adds the files `~/.ssh/id_rsa`, `~/.ssh/id_dsa`, and `~/.ssh/identity`. Alternative file names can be given on the command line, or identities can be gathered from the user's key ring (real or virtual). To obtain them from SAF key rings, use either the _ZOS_SSH_KEY_RING or _ZOS_SSH_KEY_RING_LABEL environment variables. For more information about them, see "Environment variables" on page 101.

Tip: Users sharing a UNIX UID should always run **ssh-add** with arguments to specify the identities to be added or removed. If any file requires a passphrase, **ssh-add** asks for the passphrase from the user. The passphrase is read from the user's tty. **ssh-add** retries the last passphrase if multiple identity files are given.

Requirement: The authentication agent must be running and the SSH_AUTH_SOCK environment variable must contain the name of its socket for **ssh-add** to work.

Options

–c Specifies that added identities are subject to confirmation by the SSH_ASKPASS program before being used for authentication. You can press Enter or type 'yes' to confirm use of the identities. The SSH_ASKPASS program is described in “Environment variables” on page 101.

–d Removes the identity from the agent. When run without specifying an identity to remove, it removes `~/.ssh/id_rsa`, `~/.ssh/id_dsa`, and `~/.ssh/identity`. If the default identities are not present, **ssh-add** ends with return code 1.

When the identity is specified, **ssh-add** needs to load the public key of the identity first in order to remove it. It looks for the public key in the path name of the identity. If the key is not found, the error message “Bad key file” is given.

–D Deletes all identities from the agent.

-e *reader* Not supported in z/OS UNIX. Removes key in the smart card reader.

–l Lists fingerprints of all identities currently represented by the agent.

–L Lists public key parameters of all identities currently represented by the agent.

-s *reader* Not supported in z/OS UNIX. Adds key in smart card reader.

–t *life* Sets a maximum lifetime when adding identities to an agent. The lifetime can be specified in seconds or in a time format specified in **sshd_config**.

–x Locks the agent with a password.

–X Unlocks the agent.

Files

~/.ssh/identity
Contains the protocol version 1 RSA authentication identity of the user.

~/.ssh/id_dsa
Contains the protocol version 2 DSA authentication identity of the user.

~/.ssh/id_rsa
Contains the protocol version 2 RSA authentication identity of the user.

Identity files should not be readable by anyone but the user. **ssh-add** ignores identity files if they are accessible by others.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SSH_KEY_RING
Specifies the key ring owner, followed by that user's SAF key ring name to be used as input, rather than the default or specified file names. The owner and key ring name must be separated by a '/'. All RSA and DSA identities that are in this key ring will be added to the authentication agent. The key ring can be either real or virtual.

Example:

```
KeyRingOwner/KeyRingName
```

If both _ZOS_SSH_KEY_RING and _ZOS_SSH_KEY_RING_LABEL are set, then only _ZOS_SSH_KEY_RING_LABEL is used.

_ZOS_SSH_KEY_RING_LABEL
Specifies the key ring owner, followed by that user's SAF key ring and certificate label within the key ring containing the input key, rather than the default or specified file names. The owner and key ring name must be separated by a '/'. One or more blanks separate the key ring name from the certificate label. Labels can contain embedded blanks. When setting the variable on a shell command line, the value must be enclosed in double quotes to preserve the blanks. The key ring can be either real or virtual.

Example:

```
KeyRingOwner/KeyRingName CertLabel
```

If both _ZOS_SSH_KEY_RING and _ZOS_SSH_KEY_RING_LABEL are set, then only _ZOS_SSH_KEY_RING_LABEL is used.

DISPLAY, SSH_ASKPASS
If **ssh-add** needs a passphrase, it will read the passphrase from the current terminal if it was run from a terminal. If **ssh-add** does not have a terminal associated with it, but DISPLAY and SSH_ASKPASS are set, it will execute the program specified by SSH_ASKPASS and open an X11 window to read the passphrase. This is particularly useful when calling **ssh-add** from an .Xsession or a script. It is necessary to redirect the input from `/dev/null` to make this work.

Example:

```
ssh-add < /dev/null
```

SSH_AUTH_SOCK
Identifies the path of a UNIX-domain socket used to communicate with the agent.

Exit values

0 Successful completion
1 An error occurred. The specified command failed.
2 An error occurred. **ssh-add** is unable to contact the authentication agent.

Related information

ssh, **ssh-agent**, **ssh-keygen**, **sshd**

Authors

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

---

ssh-agent — Authentication agent

Format

ssh-agent [**–c** | **–s**] [**–d**] [**–a** *bind_address*] [**–t** *life*] [*command_string* [*args ...*]]
ssh-agent [**–c** | **–s**] **–k**

Description

ssh-agent is a program to hold private keys used for public key authentication (RSA, DSA). The idea is that **ssh-agent** is started in the beginning of an X-session or a login session and all other windows or programs are started as clients to the **ssh-agent** program. Through the use of environment variables, the agent can be located and automatically used for authentication when logging in to other machines using **ssh**.

The agent initially does not have any private keys. Keys are added using **ssh-add**. When executed without arguments, **ssh-add** adds the files `~/.ssh/id_rsa`, `~/.ssh/id_dsa`, and `~/.ssh/identity`. If the identity has a passphrase, **ssh-add** asks for the passphrase (using a small X11 application if running under X11 or from the terminal if running without X11). It then sends the identity to the agent. Several identities can be stored in the agent; the agent can automatically use any of these identities. **ssh-add –l** displays the identities currently held by the agent. Identities stored in the agent will take precedence over an identity specified through **ssh**'s **–i** option or IdentityFile keyword. Refer to the **-i** *identity_file* description in “ssh” on page 85 for a summary of the order that identities are tried during public key authentication.

The concept is that the agent run is in the user's local machine. Authentication data need not be stored on any other machine and authentication passphrases never go over the network. However, the connection to the agent is forwarded over SSH remote logins and the user can thus use the privileges given by the identities anywhere in the network in a secure way.

There are two main ways to set up an agent. Either the agent starts a new subcommand into which some environment variables are exported or the agent prints the needed shell commands (either **sh** or **tcsh** syntax can be generated) which can be run with **eval** in the calling shell. Later, **ssh** looks at these variables and uses them to establish an agent. The agent will never send a private key over its request channel. Instead, operations that require a private key will be performed by the agent and the result will be returned to the requester. This way, private keys are not exposed to clients using the agent. For example:

For the **sh** syntax:

1. `ssh-agent $SHELL`

2. `eval 'ssh-agent -s'`

For **tcsh** syntax:

1. `ssh-agent $SHELL`
2. `eval 'ssh-agent -c'`

A UNIX-domain socket is created and the name of this socket is stored in the SSH_AUTH_SOCK environment variable. The socket is owned by the current user and is thereby accessible to processes running under the same user ID and superusers.

The SSH_AGENT_PID environment variable holds the agent's process ID. The agent exits automatically when the command given on the command line terminates.

Options

–a *bind_address* Binds the agent to the UNIX-domain socket *bind_address*. The default is `/tmp/ssh-XXXXXXXX/agent.<ppid>`.

–c Generates C-shell (**tcsh**) commands on stdout. This is the default if SHELL looks like it is a csh style of shell.

–d Debug mode. When this option is specified, **ssh-agent** will not fork.

–k Kills the current agent given by the SSH_AGENT_PID environment variable). This is only necessary when **ssh-agent** is run with **eval** in the calling shell. If the agent started a new subshell then exiting the subshell will also kill the agent.

–s Generates Bourne shell (**sh**) commands on stdout. This is the default if SHELL does not look like it is a csh style of shell.

–t *life* Sets a default value for the maximum lifetime of identities added to the agent. The lifetime can be specified in seconds or in a time format specified in **sshd**. A lifetime specified for an identity with **ssh-add** overrides this value. Without this option, the default maximum lifetime is forever.

If a *command_string* is given, this is executed as a subprocess of the agent. When the command ends, so does the agent.

Files

~/.ssh/identity
Contains the protocol version 1 RSA authentication identity of the user.

~/.ssh/id_dsa
Contains the protocol version 2 DSA authentication identity of the user.

~/.ssh/id_rsa
Contains the protocol version 2 RSA authentication identity of the user.

/tmp/ssh-XXXXXXXXXX/agent.<ppid>
UNIX-domain sockets used to contain the connection to the authentication agent. **ppid** is the process ID of the agent's parent process. The last eight characters of “XXXXXXXXXX ” will match ppid if the ppid is eight characters. Otherwise, “XXXXXXXXXX” is a system-generated string. These

sockets should be readable only by the owner. The sockets should be automatically removed when the agent exits.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

SHELL
Contains the full path name of the current shell.

SSH_AGENT_PID
Holds the process ID of the agent.

SSH_AUTH_SOCK
Holds the name of the socket through which the agent is accessible.

Exit values

0 Successful completion
> 0 Failure

Related information

ssh, **ssh-add**, **ssh-keygen**, **sshd**

Authors

OpenSSH is a derivative of the original and free **ssh** 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

---

ssh-askpass — X11-based passphrase dialog for OpenSSH

Description

ssh-askpass is an X11-based passphrase dialog for use with OpenSSH. It is intended to be called from the **ssh-add** program and not invoked directly.

The user interface has a series of LED-like areas which light up one-by-one with each passphrase character entered, beginning from the left-hand edge of the dialog. When they reach the right hand edge, they go dark one-by-one again. This gives the user feedback that passphrase characters have been entered, but does not provide onlookers with a cue as to the length of the passphrase.

Pressing the OK button accepts the passphrase (even if it is empty), which is written to standard output and the dialog exits with a status of zero (success). Pressing the Cancel button discards the passphrase and the dialog exits with nonzero status.

The following keystrokes are accepted:

[Backspace] or [Delete]
Erases previous character

[Control+U] or [Control+X]
Erases entire passphrase

[Enter], [Control+M], or [Control+J]
Accepts passphrase (OK)

[Escape]
Discards passphrase (Cancel)

Files

/usr/lib/X11/app-defaults
The definition and files for **x11-ssh-askpass** are available at http://www.jmknoble.net/software/x11-ssh-askpass/.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

Exit values

0 Successful completion
> 0 Bad passphrase entered or an error occurred

Related information

ssh, **ssh-add**, **sshd**

Authors

Jamie Zawinski, Jim Knoble

---

ssh-keygen — Authentication key generation, management, and conversion

Format

ssh-keygen [**-q**] [**-b** *bits*] [**-t** *type*] [**-N** *new_passphrase*] [**-C** *comment*] [**-f** *output_keyfile*]

ssh-keygen –p [**–P** *old_passphrase*] [**-N** *new_passphrase*] [**–f** *keyfile*]

ssh-keygen –i [**–f** *input_keyfile*]

ssh-keygen –e [**–f** *input_keyfile*]

ssh-keygen –y [**–f** *input_keyfile*]

ssh-keygen –c [**–P** *passphrase*] [**–C** *comment*] [**–f** *keyfile*]

ssh-keygen –l [**–f** *input_keyfile*]

ssh-keygen –B [**–f** *input_keyfile*]

ssh-keygen –F *hostname* [**–f** *known_hosts_file*] [**–H**]

ssh-keygen –H [**–f** *known_hosts_file*]

ssh-keygen –R *hostname* [**–-f** *known_hosts_file*]

ssh-keygen –r *hostname* [**–f** *input_keyfile*] [**–g**]

ssh-keygen –G *output_file* [**–v**] [**–b** *bits*] [**–M** *memory*] [**–S** *start_point*]
ssh-keygen –T *output_file* [**–f** *input_file*] [**–v**] [**–a** *num_trials*] [**–W** *generator*]

Description

ssh-keygen generates, manages, and converts authentication keys for **ssh**. It can create RSA keys for use by SSH protocol version 1 and RSA or DSA keys for use by SSH protocol version 2. The type of key to be generated is specified with the **–t** option. If invoked without any arguments, **ssh-keygen** generates an RSA key for use in SSH protocol 2 connections.

ssh-keygen supports the extraction and conversion of keys that are stored in digital certificates associated with SAF key rings.

ssh-keygen is also used to generate groups for use in Diffie-Hellman Group Exchange (DH-GEX). It is a key agreement method that allows two parties to derive a shared secret key securely over an open (unprotected) network. For more details, check the IETF Internet draft "Diffie-Hellman Group Exchange for the SSH Transport Layer Protocol" at http://www.ietf.org/rfc/rfc4253.txt. For additional information, see "Moduli generation" on page 109.

If not using SAF key rings, each user who wants to use SSH with RSA or DSA authentication runs **ssh-keygen** once to create the authentication key in `~/.ssh/identity`, `~/.ssh/id_dsa`, or `~/.ssh/id_rsa`. The system administrator might also use **ssh-keygen** to generate host keys.

This program generates the key and asks for a file in which to store the private key. The public key is stored in a file with the same name but with ".pub" appended. The program also asks for a passphrase. A passphrase is similar to a password, except it can be a phrase with a series of words, punctuation, numbers, white space, or any string of characters you want. Unless it is empty, the passphrase must be greater than 4 characters long. However, good passphrases are 10 to 30 characters long, are not simple sentences or otherwise guessable (English prose has only 1 or 2 bits of entropy per character and provides very bad passphrases), and contain a mix of uppercase and lowercase letters, numbers, and non-alphanumeric characters. The passphrase length must also be less than 1024 characters, or it will be truncated. The passphrase can be changed later using the **–p** option.

You cannot recover a lost passphrase. If the passphrase is lost or forgotten, a new key must be generated and copied to the corresponding public key to other machines.

For RSA1 keys, there is also a comment field in the key file that is only for convenience to the user to help identify the key. The comment can tell what the key is for or whatever is useful. The comment is initialized to "user@host" when the key is created, but can be changed using the **–c** option.

When a change is made to the key (such as a comment or passphrase), the change is applied to the key file only. For the loaded keys in the SSH agent, one has to unload and reload the changed keys.

When attempting to change a key, **ssh-keygen** first tries to load the key without a passphrase if one is not specified. If that fails, it will prompt for the passphrase.

Tip: To avoid problems when running as a user that shares a UID, the **-f** option can be used to specify the location of the file to process.

Options

-a *num_trials*
Specifies the number of primality tests or trials to perform when screening DH-GEX candidates using the **-T** command. The minimum number of trials is 4.

–b *bits* Specifies the number of bits in the key to create. For RSA keys, the minimum size is 768 bits, the maximum size is 32768 bits, and the default is 2048 bits. Generally, 2048 bits is considered sufficient. DSA keys must be exactly 1024 bits as specified by FIPS 186-2.

–B Shows the bubble babble digest of specified private or public key file. Bubble Babble is a text format for fingerprint. For example: `1024 xekib-ridyd-mybuh-fpun-bybir-nagak-netoc-nogib-zacev-sotim-luxex user@host.pok.ibm.com.`

–c Requests changing the comment in the private and public key files. This operation is only supported for RSA1 keys. The program will prompt for the file containing the private keys, for the passphrase if the key has one, and for the new comment, when **–P**, **–C**, and **–f** are not specified. It updates both public and private keys. This option is mutually exclusive with the **–p** option. Comments are truncated after 1023 characters. In addition, the comment length is limited by the terminal interface. For long comments up to 1023 characters, use the **–C** option.

–C *comment*
Provides the new comment. The comment is truncated after 1023 characters.

–d Specifies that the DSA type key be created. Same as the **–t** *dsa* option. It is recommended that **–t** *dsa* be used instead of **–d**.

–e Reads a private or public OpenSSH key file and prints a public key in RFC 4716 SSH Public Key File Format to stdout. This option allows exporting public keys for use by several commercial SSH implementations.

If using a SAF key ring on the local system, but not on a remote system, this option can be used with the _ZOS_SSH_KEY_RING_LABEL environment variable to export your public key from the key ring. The public key can then be copied to the remote system and imported with **ssh-keygen -i**.

Restriction: This option applies to protocol version 2 only.

–f *filename*
If **–F**, **–H**, or **–R** is specified, *filename* specifies the file name of the `known_hosts` file. For other options, *filename* specifies the file name of the key file. The *filename* is limited to 1023 characters including the 4 characters for ".pub" for the public keys.

For some of the options allowing [**-f** *input_keyfile*], the _ZOS_SSH_KEY_RING_LABEL environment variable can be used to specify a key ring and certificate label to be used, overriding the **-f** option. For more information about how the environment variable is used, see "_ZOS_SSH_KEY_RING_LABEL" on page 111.

–F *hostname*
Searches for the specified *hostname* in a `known_hosts` file, listing any

occurrences found. Use this option to find hashed host names or addresses. It can also be used in conjunction with the **–H** option to print found keys in a hashed format If **-f** is not specified, `~/.ssh/known_hosts` is used.

–g Uses generic DNS resource record format when printing fingerprint resource records using the **-r** command.

–G *output_file*
Generates candidate primes for DH-GEX.

Rule: These primes must be screened for safety (using the **-T** option) before use.

–H Hashes a `known_hosts` file. This option replaces all host names and addresses with hashed representations within the specified file; the original contents are moved to a file with a .old suffix. These hashes can be used normally by **ssh** and **sshd**, but they do not reveal identifying information if the file's contents are disclosed. This option will not modify existing hashed host names and is therefore safe to use on files that mix hashed and non-hashed names. If **-f** is not specified, `~/.ssh/known_hosts` is used.

–i Reads an unencrypted private (or public) key file in SSH protocol version 2 format and prints an OpenSSH compatible private (or public) key to stdout. **ssh-keygen** also reads the RFC 4716 SECSH Public Key File Format. This option allows importing keys from several commercial SSH implementations.

–l Shows the fingerprint of specified public key file. Private protocol version 1 RSA1 keys are also supported. For RSA and DSA keys, **ssh-keygen** tries to find the matching public key file and prints its fingerprint. For example:
```
1024 7d:74:a5:4b:7b:10:5d:62:4b:9f:f3:1c:14:32:b8:74
user@host.pok.ibm.com
```

–M *memory*
Specifies the amount of memory (in megabytes) to use when generating candidate moduli for DH-GEX. The number of specified megabytes must be an integer value greater than 7 and less than 128.

–N *new_passphrase*
Provides the new passphrase. When **–t** *type* or **–d** options are used, the **–P** value will be used for passphrase regardless if **–N** is specified. If **–P** is not specified with **–t** *type* or **–d**, the **–N** value will be used for the passphrase.

Rule: Do not specify passphrases on the command line because this method allows the passphrase to be visible (for example, when the **ps** utility is used).

–p Requests changing the passphrase of a private key file instead of creating a new private key. The program will prompt for the file containing the private key, for the old passphrase (if not empty), and twice for the new passphrase. This option is mutually exclusive with the **–c** option.

–P *passphrase*
Provides the old passphrase. When the **–t** *type* or **–d** options are used, the **–P** value is used for the passphrase regardless if **–N** is specified. When the **–t** *type* or **–d** options are used, it is recommended that **–N** *new_passphrase* be used instead of **–P** *passphrase*.

Rule: Do not specify passphrases on the command line because this method allows the passphrase to be visible (for example, when the **ps** utility is used).

–q Suppresses messages. Useful when called from a script.

–r *hostname*
Prints the SSHFP fingerprint resource record named *hostname* for the specified public key file. If **-f** is not specified, the default files `/etc/ssh/ssh_host_rsa_key` and `/etc/ssh/ssh_host_dsa_key` are used in sequence.

–R *hostname*
Removes all keys belonging to *hostname* from a `known_hosts` file. Use this option to delete hashed hosts (see the **-H** option). If **-f** is not specified, `~/.ssh/known_hosts` is used.

–S *start*
Specifies the start point in hexadecimal format when generating candidate moduli for DH-GEX. The specified start point must be a valid hexadecimal value.

–t *type* Specifies the type of the key to create. The possible values are "rsa1" for protocol version 1 and "rsa" or "dsa" for protocol version 2. The program will prompt for the file name to contain the private keys and passphrase, if **–P** or **–N**, and **–f** is not specified.

–T *output_file*
Tests Diffie-Hellman Group Exchange candidate primes (generated using the **-G** option) for safety.

–U *reader*
Not supported in z/OS UNIX. Uploads an existing RSA private key into the smart card in reader.

–v Verbose mode. Causes **ssh-keygen** to print debugging messages about its progress. The messages are helpful for debugging moduli generation. Multiple **–v** options increase the verbosity. You can specify up to three **-v** options.

–W *generator*
Specifies the desired generator when testing candidate module for DH-GEX. Valid generator values are 2, 3, or 5.

–x Same as **–e**. It is recommended that **–e** be used instead of **–x**.

–X Same as **–i**. It is recommended that **–i** be used instead of **–X**.

–y Reads a private OpenSSH format file and prints an OpenSSH public key to stdout.

Moduli generation

You can use **ssh-keygen** to generate groups for the Diffie-Hellman Group Exchange (DH-GEX) protocol. DH-GEX is a key agreement method that allows two parties to derive a shared secret key securely over an open (unprotected) network.

Generating these groups is a two-step process. First, candidate primes are generated using a fast, but memory-intensive process. These candidate primes are then tested for suitability, which is a CPU-intensive process.

Use the **-G** option to generate the primes. You can specify the length of the primes using the **-b** option.

Example:

```
ssh-keygen -G moduli-2048.candidates -b 2048
```

By default, the search for primes begins at a random point in the desired length range. You can override this using the **-S** option, which specifies a different start point (in hex).

After a set of candidates has been generated, the candidates must be tested for suitability using the **-T** option. In this mode, **ssh-keygen** reads the candidates from standard input (or a file specified using the **-f** option).

Example:

```
ssh-keygen -T moduli-2048 -f moduli-2048.candidates
```

By default, each candidate is subject to 100 primality tests. You can override the default by using the **-a** option. The DH generator value is automatically chosen for the prime under consideration. If you want a specific generator, you can request it using the **-W** option. Valid generator values are 2, 3 and 5.

You can install screened DH groups in `/etc/ssh/moduli`.

Requirement: The `/etc/ssh/moduli` file must contain moduli of a range of bit lengths, and both ends of a connection must share common moduli.

Files

/etc/ssh/moduli
Contains Diffie-Hellman groups used for DH-GEX. The file format is described in "moduli" on page 160.

~/.ssh/identity
Contains the protocol version 1 RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 3DES. This file is not automatically accessed by **ssh-keygen**, but it is offered as the default file for the private key. **ssh** reads this file when a login attempt is made.

~/.ssh/identity.pub
Contains the protocol version 1 RSA public key for authentication. The contents of this file should be added to the `~/.ssh/authorized_keys` file on all machines where the user wants to log in using RSA authentication. You do not need to keep the contents of this file secret.

~/.ssh/id_dsa
Contains the protocol version 2 DSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 3DES. This file is not automatically accessed by **ssh-keygen**, but it is offered as the default file for the private key. **ssh** reads this file when a login attempt is made.

~/.ssh/id_dsa.pub
Contains the protocol version 2 DSA public key for authentication. The contents of this file should be added to the `~/.ssh/authorized_keys` file on all machines where the user wants to log in using DSA authentication. You do not need to keep the contents of this file a secret.

~/.ssh/id_rsa
Contains the protocol version 2 RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to

specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 3DES. This file is not automatically accessed by **ssh-keygen**, but it is offered as the default file for the private key. **ssh** reads this file when a login attempt is made.

~/.ssh/id_rsa.pub
Contains the protocol version 2 RSA public key for authentication. The contents of this file should be added to `~/.ssh/authorized_keys` on all machines where the user wants to log in using RSA authentication. You do not need to keep the contents of this file secret.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SSH_KEY_RING_LABEL
Specifies the key ring owner, followed by that user's SAF key ring and certificate label within the key ring containing the input key, rather than the file specified as **-f** *input_keyfile*, on some **ssh-keygen** options. The key ring owner and key ring name must be separated by a '/'. One or more blanks separate the key ring name from the certificate label. Labels can contain embedded blanks. When setting the variable on a shell command line, the value must be enclosed in double quotes to preserve the blanks.

Example:

```
KeyRingOwner/KeyRingName CertLabel
```

The key ring can be either real or virtual.

This variable is used on the following options: **-e**, **-l**, **-r**, **-y**, and **-B**. Other options that use the **-f** *input_keyfile* will ignore this variable.

Exit values

0 Successful completion
> 0 Failure

Related information

ssh, **ssh-add**, **ssh-agent**, **sshd**

Authors

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

ssh-keyscan — Gather ssh public keys

Format

ssh-keyscan [**–46Hv**] [**–f** *file*] [**–p** *port*] [**–T** *timeout*] [**–t** *type*] [*host* | *addrlist namelist*] [...]

Description

ssh-keyscan is a command for gathering the public host keys for a number of hosts. It aids in building and verifying `ssh_known_hosts` files. **ssh-keyscan** provides a minimal interface suitable for use by shell and Perl scripts.

ssh-keyscan uses non-blocking socket I/O to contact as many hosts as possible in parallel, so it is very efficient. For successful host key collection, you do not need login access to the machines that are being scanned, nor does the scanning process involve any encryption.

If a machine being scanned is down or is not running **sshd**, the public key information cannot be collected for that machine. The return value is not altered and a warning message might be displayed.

Example:

```
ssh-keyscan hostname1 hostname2
hostname1: exception!
(hostname2's rsa1 key displayed here)
```

Options

–4 Forces **ssh-keyscan** to use IPv4 addresses only. If both **–4** and **–6** are specified, **ssh-keyscan** uses the option that appears last on the command line.

–6 Forces **ssh-keyscan** to use IPv6 addresses only. If both **–4** and **–6** are specified, **ssh-keyscan** uses the option that appears last on the command line.

–f *file* Reads *hosts* or *addrlist namelist* pairs from this file, one per line. If **–** is supplied instead of a file name, **ssh-keyscan** reads *hosts* or *addrlist namelist* pairs from the standard input.

–H Hashes all host names and addresses in the output. Hashed names can be used normally by **ssh** and **sshd**, but they do not reveal identifying information if the host's contents are disclosed.

–p *port*
Port to connect to on the remote host.

–t *type* Specifies the type of the key to fetch from the scanned hosts. The possible values are "rsa1" for protocol version 1 and "rsa" or "dsa" for protocol version 2. If the **-t** option is not specified, **ssh-keyscan** searches only for SSH protocol version 1 keys ("rsa1") by default. If the target machine does not support SSH protocol version 1, then nothing is returned or displayed for that machine

–T *timeout*
Sets the timeout for connection attempts. If timeout seconds have elapsed since a connection was initiated to a host or since the last time anything was read from that host, then the connection is closed and the host in question considered unavailable. The default is 5 seconds.

–v Verbose mode. Causes **ssh-keyscan** to print debugging messages about its progress. Multiple **-v** options increase the verbosity. You can specify up to three **-v** options.

File formats

Input format

Each line of the input file shall consist of either *hosts* or *addrlist namelist* pairs. *Hosts* is either a single or comma-delimited list of hosts. *Addrlist* is a single or comma-separated list of IP addresses and *namelist* is either a single or comma-delimited list of hosts. *Addrlist namelist* pairs are separated by white space.

Example: Examples of input file lines:

```
1.2.3.4
name.my.domain
1.2.3.4,1.2.4.4
1.2.3.4,1.2.4.4 name.my.domain,name,n.my.domain,n
name.my.domain,1.2.3.4,name,n,1.2.4.4,n.my.domain
```

Output format for rsa1 keys

host-or-namelist bits exponent modulus

Output format for rsa and dsa keys

host-or-namelist keytype base64-encoded-key where keytype is either *ssh-rsa* for an RSA key or *ssh-dss* for a DSA key

Files

/etc/ssh/ssh_known_hosts

System-wide list of known host keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization. See "ssh_known_hosts file format" on page 122 for further details of the format of this file. This file must be writeable only by the owner and only be world-readable.

Environment variables

_ZOS_OPENSSH_DEBUG

Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT

Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

Exit values

0 Successful completion

> 0 An error occurred

Usage note

ssh-keyscan generates "Connection closed by remote host" messages on the consoles of all the machines it scans if the server is older than version 2.9. The connection is closed because it opens a connection to the **ssh** port, reads the public key, and drops the connection as soon as it gets the key.

Related information

ssh, **sshd**

Authors

David Mazieres wrote the initial version, and Wayne Davison added support for protocol version 2.

---

ssh-keysign — ssh helper program for host-based authentication

Format

ssh-keysign

Description

ssh-keysign is used by **ssh** to access the local host keys and generate the digital signature that is required during host-based authentication with SSH protocol version 2. **ssh-keysign** is not intended to be invoked by the user, but from **ssh**. See “ssh” on page 85 and “sshd” on page 116 for more information about host-based authentication.

ssh-keysign is disabled by default. It can only be enabled in the global client configuration file `/etc/ssh/ssh_config` by setting EnableSSHKeysign to "yes".

Files

/etc/ssh/ssh_config
Controls whether **ssh-keysign** is enabled. EnableSSHKeysign must be set to “yes” in this file.

/etc/ssh/ssh_host_dsa_key, /etc/ssh/ssh_host_rsa_key
These files contain the private parts of the host keys used to generate the digital signature. They should be owned by a superuser, readable only by a superuser, and not accessible by others.

Restriction: Because they are readable only by UID 0, **ssh-keysign** must be setuid 0 if host-based authentication is used.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

Exit values

0 Successful completion

> 0 An error occurred

Related information

ssh, **ssh-keygen**, **ssh_config**, **sshd**

Authors

Markus Friedl

ssh-rand-helper — Gather random numbers for OpenSSH

Format

ssh-rand-helper [**–hvxX**] [**–b** *bytes*]

Description

ssh-rand-helper is a small helper program used by **ssh**, **ssh-add**, **ssh-agent**, **ssh-keygen**, **ssh-keyscan**, **ssh-keysign**, and **sshd** to gather random numbers of cryptographic quality.

Typically, **ssh-rand-helper** generates a strong random seed and provides it to the calling program via standard output. If standard output is a tty, **ssh-rand-helper** instead prints the seed in hexadecimal format unless told otherwise.

By default, **ssh-rand-helper** gathers random numbers from the commands listed in `/etc/ssh/ssh_prng_cmds`. The output of each of the commands listed is hashed and used to generate a random seed for the calling program. The _ZOS_SSH_PRNG_CMDS_TIMEOUT environment variable can be used to control the timeout value when running a command. **ssh-rand-helper** also stores seed files in `~/.ssh/prng_seed` between executions.

Options

This program is not intended to be run by the user, so the few command-line options are for debugging purposes only.

–b *bytes* Specifies the number of random bytes to include in the output.

–h Displays a summary of options.

–v Turns on debugging messages. Multiple **–v** options increase the debugging level. You can specify up to three **-v** options.

–x Specifies that seeds are to be output in hexadecimal format instead of binary format.

–X Forces output of a binary seed, even if standard output is a tty.

Files

/etc/ssh/ssh_prng_cmds
Contains the system commands used to generate random data. This file can be modified by a system administrator to control the trade-off between the level of randomness and performance.

~/.ssh/prng_seed
Seed file used by **ssh-rand-helper**.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SSH_PRNG_CMDS_TIMEOUT
The timeout value used by **ssh-rand-helper** when running a command from the `/etc/ssh/ssh_prng_cmds` file. The timeout value is in milliseconds and has a minimum value of 1 and a maximum value of 2147483646. The default value is 1000. To determine if the software algorithm **ssh-rand-helper** is being used instead of hardware support to generate a random seed for an OpenSSH command, see "Verifying if hardware support is being used" on page 50.

Exit values

0 Successful completion

> 0 An error occurred.

Related information

ssh, **ssh-add**, **ssh-keygen**, **sshd**

Author

Damien Miller

---

sshd — OpenSSH daemon

Format

sshd [**-46dDeiqt**] [**–b** *bits*] [**–f** *config_file*] [**–g** *login_grace_time*] [**–h** *host_key_file*] [**–k** *key_gen_time*] [**–o** *option*] [**–p** *port*] [**–u** *len*]

Description

sshd (OpenSSH daemon) is the daemon program for **ssh**. Together, these programs are an alternative to **rlogin** and **rsh** and provide encrypted communications between two untrusted hosts over an insecure network.

sshd listens for connections from clients. It is typically started when z/OS UNIX is initialized. (See Chapter 5, "For system administrators," on page 21 for more information about starting **sshd**.) It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange. This implementation of **sshd** supports both SSH protocol versions 1 and 2 simultaneously. The default **sshd** configuration only runs protocol version 2

OpenSSH can be configured to collect SMF login failure records for **sshd** as well as server transfer completion records that are associated with "internal-sftp". See "Steps for setting up the system to collect OpenSSH SMF records" on page 51 for more information. See Chapter 12, "SMF Type 119 records for OpenSSH," on page 167 for more information about the SMF login failure records (subtype 98) and server transfer completion records (subtype 96).

Options

sshd can be configured using command-line options or a **sshd_config** configuration file (the default is **/etc/ssh/sshd_config**); command-line options override values specified in the configuration file. **sshd** can also obtain z/OS-specific configuration data from a system-wide **/etc/ssh/sshd_config** configuration file.

sshd rereads its configuration files, including z/OS-specific files, when it receives a hang up signal, SIGHUP, by executing itself with the name and options it was started with; for example, `/usr/sbin/sshd`.

For more information about the configuration files, see "sshd_config" on page 144 and "zos_sshd_config" on page 158.

–4 Forces **sshd** to use IPv4 addresses only. If both **–4** and **–6** are specified, **sshd** uses the option that appears last on the command line.

–6 Forces **sshd** to use IPv6 addresses only. If both **–4** and **–6** are specified, **sshd** uses the option that appears last on the command line.

–b *bits* Specifies the number of bits in the ephemeral protocol version 1 server key (default 768).

–d Debug mode. The server sends verbose debug output to the system log (if **sshd** is invoked with **–i**) or stderr, and does not put itself in the background. The server also will not fork and will only process one connection. This option is only intended for debugging for the server. Multiple **–d** options increase the debugging level. You can specify up to three **-v** options.

–D **sshd** does not fork and does not become a daemon. This allows for easy monitoring of **sshd**.

–e **sshd** sends the output to standard error instead of the system log. This option is only useful when **sshd** is not running as a daemon (for example, when **sshd** is started with the **–D** option).

–f *config_file*
Specifies the name of the **sshd_config** configuration file. The default is `/etc/ssh/sshd_config`. **sshd** will not start if there is no **sshd_config** configuration file. This option has no effect on the z/OS-specific configuration file.

–g *login_grace_time*
Gives the grace time for clients to authenticate themselves (default 120 seconds). If the client fails to authenticate the user within this many seconds, the server disconnects and exits. A value of zero indicates no limit.

–h *host_key_file*
Specifies a file from which a host key is read.

If **sshd** is not run as UID(0), a host key must often be provided by another method because the default host key files are normally not readable by anyone but a superuser. Host keys can be provided by either using this option or by specifying a host key with either the HostKey or HostKeyRingLabel configuration options. For full details of the options and their values, see "sshd_config" on page 144 and "zos_sshd_config" on page 158.

The default host key file is `/etc/ssh/ssh_host_key` for protocol version 1. For protocol version 2, the default host key files are `/etc/ssh/ssh_host_rsa_key` and `/etc/ssh/ssh_host_dsa_key`. It is possible to have multiple host keys for the different protocol versions and host key algorithms.

–i Specifies that **sshd** is being run from **inetd**. **sshd** is normally not run from **inetd** because it needs to generate the server key before it can respond to the client and this might decrease performance. Clients would have to wait

too long if the key was regenerated every time. However, with small key sizes (such as 512), using **sshd** from **inetd** might be feasible.

–k *key_gen_time*
Specifies how often the ephemeral protocol version 1 server key is regenerated (default 3600 seconds or one hour). The motivation for regenerating the key fairly often is that the key is not stored anywhere, and after about an hour, it becomes impossible to recover the key for decrypting intercepted communications even if the machine is cracked into or physically seized. A value of zero indicates that the key will never be regenerated. The key will only be regenerated if it has been used.

–o *option*
Can be used to give options in the format used in the **sshd_config** and **zos_sshd_config** configuration files. This is useful for specifying options for which there is no separate command-line flag. For full details of the options and their values, see "sshd_config" on page 144 and "zos_sshd_config" on page 158.

–p *port*
Specifies the port on which the server listens for connections (default 22). Multiple port options are permitted. Ports specified in the **sshd_config** configuration file with the Port option are ignored when a command-line port is specified. Ports specified using the ListenAddress option override command-line ports. More information about those options can be found in "Port" on page 155 and "ListenAddress" on page 152.

–q Quiet mode. Nothing is sent to the system log. Typically, the beginning, authentication, and termination of each connection is logged.

–t Test mode. Only checks the validity of the **sshd_config** configuration file and sanity of the keys. This option is useful for updating **sshd** reliably because configuration options might change.

–u *len* This option is used to specify the size of the field in the utmpx structure that holds the remote host name. If the resolved host name is longer than *len*, the dotted decimal value will be used instead. This allows hosts with very long host names that overflow this field to still be uniquely identified. Specifying **–u0** indicates that only dotted decimal addresses should be put into the utmpx file. **–u0** can also be used to prevent **sshd** from making DNS requests unless the authentication mechanism or configuration requires it. Authentication mechanisms that might require DNS include Rhostsauthentication, RhostsRSAAuthentication, HostbasedAuthentication, and using a *from="pattern-list"* option in a key file. Configuration options that require DNS include using a *user@host* pattern in AllowUsers or DenyUsers.

Authentication

The OpenSSH SSH daemon supports SSH protocols versions 1 and 2. Protocol version 2 is supported by default, though this can be changed by using the Protocol keyword in **sshd_config**. (The keyword is described in "Protocol" on page 155.) Protocol version 2 supports both RSA and DSA keys; protocol version 1 only supports RSA keys. For both protocols, each host has a host-specific key used to identify the host.

Forward security for protocol version 1 is provided through an additional server key that is generated when the server starts. This key is typically regenerated every hour if it has been used, and is never stored on disk. Whenever a client

connects, the daemon responds with its public host and server keys. The client compares the RSA host key against its own database to verify that it has not changed. The client then generates a 256-bit random number. It encrypts this random number using both the host key and the server key, and sends the encrypted number to the server. Both sides then use this random number as a session key which is used to encrypt all further communications in the session. The rest of the session is encrypted using a conventional cipher, currently Blowfish or 3DES, with 3DES being used by default. The client selects the encryption algorithm to use from those offered by the server.

For protocol version 2, forward security is provided through a Diffie-Hellman key agreement. This key agreement results in a shared session key. The rest of the session is encrypted using a symmetric cipher. The client selects the encryption algorithm to use from those offered by the server. For a list of ciphers, see "Ciphers" on page 147. Additionally, session integrity is provided through a cryptographic message authentication code. For a list of MACs keywords, see "MACs" on page 152.

Finally, the server and the client enter an authentication dialog. The client tries to authenticate itself using host-based authentication (which is disabled by default), public key authentication, challenge-response authentication (which is not supported on z/OS UNIX), or password authentication.

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time the client can request tasks such as allocating a pty, forwarding X11 connections, forwarding TCP connections, or forwarding the authentication agent connection over the secure channel.

After this, the client either requests a shell or execution of a command. The sides then enter session mode. In this mode, either side can send data at any time, and such data is forwarded to and from the shell or command on the server side, and the user terminal on the client side.

When the user program terminates and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.

Login process

When a user successfully logs in and privilege separation is disabled, **sshd** goes through the following series of steps. If privilege is enabled, then Step 4 is done first, and then Steps 1, 2, 3, 5, 6, 7, 8, and 9 in that order. As a result, the `/etc/motd`, `/etc/nologin` and `~/.hushlogin` path names are relative to the user's new root directory.

1. If the login is on a tty and no command has been specified, prints last login time and `/etc/motd` (unless prevented in the configuration file or by `~/.hushlogin`; see "Files" on page 124 for details).
2. If the login is on a tty, records login time to the utmpx database.
3. If the user is not a superuser, checks `/etc/nologin`; if it exists, prints contents and quits.
4. Changes to run with normal user privileges. The changes include processing the **sshd_config** ChrootDirectory keyword. As a result, path name processing after this point is relative to the user's new root directory. The keyword is described in "ChrootDirectory" on page 146.
5. Sets up basic environment.

6. Reads the ~/.ssh/environment file if it exists and if users are allowed to change their environment. See the keyword "PermitUserEnvironment" on page 154.
7. Changes to the user's home directory.
8. If the ~/.ssh/rc file exists, runs it; or, if /etc/ssh/sshrc exists, runs it; otherwise runs the xauth program. The rc files are given the X11 authentication protocol and cookie in standard input. This method of reading only the first startup file found differs from that of the z/OS shells.
9. Runs the user's shell or command.

Format of the authorized_keys file

The AuthorizedKeysFile keyword specifies the file containing public keys for public key authentication. If none is specified, the default is ~/.ssh/authorized_keys.

Each line of the file contains one key specification (empty lines and lines starting with # are ignored as comments).

- Protocol version 1 public keys consist of the following space-separated fields: options, bits, exponent, modulus, comment. The bits, exponent, modulus, and comment fields give the RSA key for protocol version 1.
- Protocol version 2 public keys that are not in key rings consist of options, keytype, base64-encoded key, comment. The options field is optional; its presence is determined by whether the line starts with a number (the options field never starts with a number).

  Protocol version 2 public keys that are in a key ring only consist of options, one of which must be the *zos-key-ring-label* option.

  For protocol version 2, the keytype is "ssh-dss" or "ssh-rsa".

Lines in this file are typically several hundred bytes long (because of the size of the public key encoding) up to a limit of 8 kilobytes, which permits DSA keys up to 8 kilobits and RSA keys up to 16 kilobits. To avoid typing them, copy the identity.pub, id_dsa.pub, or id_rsa.pub file and edit it.

sshd enforces a minimum RSA key modulus size for protocol version 1 and protocol version 2 keys of 768 bits.

The options field (if present) consists of comma-separated option specifications. No spaces are permitted, except within double quotes. The following option specifications are supported (note that option keywords are not case sensitive):

command="command"
: Specifies that the command is executed whenever this key is used for authentication. The command supplied by the user (if any) is ignored. The command is on a pseudo terminal (pty) if the client requests a pty; otherwise it is run without a tty. If an 8-bit clean channel is required, do not request a pty or should specify no-pty. A quote can be included in the command by quoting it with a backslash. This option can be useful to restrict certain public keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. The client can specify any combination of TCP and X11 forwarding unless they are explicitly prohibited. The command originally supplied by the client is available in the SSH_ORIGINAL_COMMAND environment variable. This option applies to shell, command, or subsystem execution.

environment="NAME=value"
: Specifies that the string is to be added to the environment when logging in

using this key. Environment variables set this way override other default environment values. See “Environment variables” on page 98 in **ssh** for more information. Multiple options of this type are permitted. Environment processing is disabled by default and is controlled by means of the PermitUserEnvironment option. This option is automatically disabled if UseLogin is enabled.

See “PermitUserEnvironment” on page 154 for information about environment variable processing and precedence rules. The **sshd_config** keyword UseLogin is documented in “UseLogin” on page 157.

from="pattern-list"
Specifies that in addition to public key authentication, the canonical name of the remote host must be present in the comma-separated list of patterns. The purpose of this option is to increase security; public key authentication by itself does not trust the network or name servers or anything but the key. However, if the key is stolen, this additional option makes using a stolen key more difficult because name servers and routers would have to be compromised in addition to just the key.

See “Patterns” on page 140 for more information about patterns.

no-agent-forwarding
Prevents authentication agent forwarding when this key is used for authentication.

no-port-forwarding
Prevents TCP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This option can be used in conjunction with the command option.

no-pty Prevents tty allocation (a request to allocate a pty will fail).

no-user-rc
Disables execution of the `~/.ssh/rc` file.

no-X11-forwarding
Prevents X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

permitopen="host:port"
Limits local **ssh –L** port forwarding such that it can only connect to the specified host and port. IPv6 addresses can be specified with an alternate syntax: *host/port*. Use commas to separate multiple permitopen options. No pattern matching is performed on the specified host names.

Restriction: The maximum number of permitted opens is 100.

Appendix B, “OpenSSH - port forwarding examples,” on page 335 has examples of port forwarding.

tunnel="n"
This option is ignored on z/OS UNIX. Forces a tunnel device on the server. Without this option, the next available device is used if the client requests a tunnel.

zos-key-ring-label=**"***KeyRingOwner/KeyRingName label***"**
Specifies the key ring owner, key ring name, and the certificate label within the key ring on the OpenSSH server that contains the user's public key. One or more blanks separate the key ring (real or virtual) name from the

certificate label. Certificate labels can contain embedded blanks. The option value must be enclosed in double quotes. Key fields following the options (on the same line) are ignored.

Requirements:

- The certificate must be copied from the client system and added to the user's key ring on the OpenSSH server.
- If the user is not storing the authorized keys in a key ring, then the public key must be extracted from the certificate and added to the user's authorized keys on the OpenSSH server.

If a key ring is being used on the server side (for example, SSHAuthKeysRing), it was created in the user authentication setup described in "Steps for setting up user authentication when keys are stored in key rings" on page 68.

An example of an `authorized_keys` file:

```
# Comments allowed at start of line
ssh-rsa AAAAB3Nza...LiPk== user@example.net
from="*.sales.example.net,!pc.sales.example.net" ssh-rsa AAAAB2...19Q== john@example.net
command="dump /home",no-pty,no-port-forwarding ssh-dss AAAAC3...51R== example.net
permitopen="192.0.2.1:80",permitopen="192.0.2.2:25" ssh-dss AAAAB5...21S==
tunnel="0",command="sh /etc/netstart tun0" ssh-rsa AAAA...==jane@example.net
zos-key-ring-label="KeyRingOwner/SSHAuthKeysRing uniq-ssh-rsa"
from="*.example.com",zos-key-ring-label="KeyRingOwner/SSHAuthKeysRing uniq-ssh-dsa"
```

ssh_known_hosts file format

The `/etc/ssh/ssh_known` hosts and `~/.ssh/known_hosts` files contain the host public keys for all known hosts. The use of the global file is optional; if it is used, it must be prepared by the administrator. The per-user file is maintained automatically. Each time the user connects from an unknown host, the key of that unknown host is added to the per-user file

Each line in these files contains the following fields, and the fields are separated by spaces:

For RSA1 from the identity.pub file:
hostnames, bits, exponent, modulus, comment.

For RSA or DSA from the id_rsa.pub or id_dsa.pub files:
hostnames, key-type, public-key, comment

For RSA or DSA from the key ring:
hostnames, zos-key-ring-label="KeyRingOwner/KeyRingName label"

zos-key-ring-label specifies the key ring owner, key ring name of the name of the `known_hosts` SAF key ring, and the certificate label of the certificate within the key ring on the OpenSSH client that contains the host public key. One or more blanks separate the key ring (real or virtual) name from the certificate label. Certificate labels can contain embedded blanks. The option value must be enclosed in double quotes. Any fields following *zos-key-ring-label* on the same line are ignored. The *zos-key-ring-label* specification keyword is not case sensitive.

Requirement: The certificate must be copied from the server system and added to the known hosts file or key ring on the OpenSSH client.

If a key ring is being used on the client side, for example, SSHKnownHostRing, the key ring was created in the server authentication setup as described in "Steps for setting up server authentication when keys are stored in key rings" on page 29.

Hostnames is a comma-separated list of patterns (* and ? act as wildcards). Each pattern is matched against the canonical host name when authenticating a client or against the user-supplied name when authenticating a server. A pattern can also be preceded by ! to indicate negation. If the host name matches a negated pattern, it is not accepted by that line even if it matched another pattern on the line. A hostname or address can optionally be enclosed within '[' and ']' brackets, then followed by ':' and a nonstandard port number.

Alternatively, hostnames can be stored in a hashed form which hides host names and addresses if the file's contents are disclosed. Hashed hostnames start with a '|' character. Only one hashed hostname can appear on a single line and none of the above negation or wildcard operators can be applied.

Bits, exponent, and modulus are taken directly from the RSA host key. They can generally be obtained from the /etc/ssh/ssh_host_key.pub file. The optional comment field continues to the end of the line.

Lines starting with # and empty lines are ignored as comments.

When performing host authentication, authentication is accepted if any matching line has the proper key. It is thus permissible (but not recommended) to have several lines or different host keys for the same names. This will happen when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information. Authentication is accepted if valid information can be found from either file.

The lines in these files are typically hundreds of characters long and should be generated by a script or by taking /etc/ssh/ssh_host_key.pub and adding the host names at the front.

An example of a ssh_known_hosts file:

```
# Comments allowed at start of line
closenet,...,192.0.2.53 1024 37 159...93 closenet.example.net
cvs.example.net,192.0.2.10 ssh-rsa AAAA1234.....=
# A hashed hostname
|1|JfKTdBh7.....= ssh-rsa AAAA1234.....=
# An example specification of a known host key from a key ring
mvs* zos-key-ring-label="KeyRingOwner/SSHKnownHostsRing mvs1-ssh-rsa"
```

Running OpenSSH in other locales

Rule: All files used by OpenSSH (such as key files and configuration files) must be in the IBM-1047 code set, with the exception of the **rc** files (/etc/ssh/sshrc and ~/.ssh/rc). The **rc** files are parsed by /bin/sh and should be in the code set of the current locale. Do not use the /etc/ssh/sshrc file if there is a possibility of the users on the system running in different locales.

Restrictions:

- OpenSSH does not run in multibyte locales.
- The OpenSSH daemon (**sshd**) must be run in the POSIX C locale (which is the default).

For more information about globalization, see Chapter 7, "Globalization on z/OS systems," on page 55.

Limitations

The maximum length of the ephemeral server key is INT_MAX.

Files

~/.hushlogin
This file is used to suppress printing the last login time and `/etc/motd`, if the **sshd_config** keywords PrintLastLog and PrintMotd, respectively, are enabled. It does not suppress printing of the banner specified by the **sshd_config** keyword Banner.

~/.rhosts

This file is used for host-based authentication. On some machines, this file might need to be world-readable if the user's home directory is on an NFS partition, because **sshd** reads it as a superuser. Additionally, this file must be owned by the user and must not have write permissions for anyone else. The recommended permission for most machines is read/write for the user and not accessible by others

~/.shosts
This file is used in exactly the same way as `~/.rhosts`, but allows host-based authentication without permitting login with **rlogin** or **rsh**.

~/.ssh/ This directory is the default location for all user-specific configuration and authentication information. There is no general requirement to keep the entire contents of this directory secret, but the recommended permissions are read/write/execute for the user, and not accessible by others.

~/.ssh/authorized_keys
Lists the public keys (RSA/DSA) that can be used for logging in as this user. For the format of this file, see "Format of the authorized_keys file" on page 120. The content of this file is not highly sensitive, but the recommended permissions are read/write for the user, and not accessible by others.

If this file, the `~/.ssh/` directory, or the user's home directory are writable by other users, then the file could be modified or replaced by unauthorized users. In this case, **sshd** will not allow it to be used unless the value for the **sshd_config** keyword StrictModes has been set to "no".

~/.ssh/environment
If this file exists, it is read into the environment at login. It can only contain empty lines, comment lines (starting with #), and assignment lines of the form *name=value*. The file must be writable only by the user; it need not be readable by anyone else. Environment processing is disabled by default and is controlled by means of the PermitUserEnvironment option, which is described in "PermitUserEnvironment" on page 154.

~/.ssh/known_hosts
Contains a list of host keys for all hosts the user has logged into that are not already in the system-wide list of known host keys, `/etc/ssh/ssh_known_hosts`. See "ssh_known_hosts file format" on page 122 for further details of the format of this file. This file must be writable only by the owner and can, but need not be, world-readable.

~/.ssh/rc

If this file exists, it is run with /bin/sh after reading the environment files, but before starting the user's shell or command. It must not produce any output on stdout; stderr must be used instead. If X forwarding is in use, it will receive the "proto cookie" pair in its standard input (and DISPLAY in its environment). The script must call xauth, because **sshd** will not run xauth automatically to add X11 cookies. If you have not configured your system for X11 forwarding, see "Steps for configuring the system for X11 forwarding" on page 47.

The primary purpose of this file is to run any initialization routines which might be needed before the user's home directory becomes accessible; AFS is a particular example of such an environment.

This file will probably contain some initialization code, followed by lines similar to this example:

```
if read proto cookie && [ -n "$DISPLAY" ]; then
     if [ `echo $DISPLAY | cut -c1-10` = 'localhost:' ]; then
           # X11UseLocalhost=yes
           echo add unix:`echo $DISPLAY |
              cut -c11-` $proto $cookie
     else
           # X11UseLocalhost=no
           echo add $DISPLAY $proto $cookie
     fi | xauth -q -
fi
```

If this file does not exist, /etc/ssh/sshrc is run, and if that does not exist either, xauth is used to add the cookie.

This file should be writable only by the user.

/etc/hosts.allow, /etc/hosts.deny

Not supported on z/OS UNIX. Access controls that should be enforced by tcp-wrappers are defined in this file.

/etc/hosts.equiv

This file is for host-based authentication. In the simplest form, this file contains host names, one per line. Users on those hosts are permitted to log in without a password, provided they have the same user name on both machines. The host name can also be followed by a user name; such users are permitted to log in as any user on this machine except superuser.

If the client host/user is successfully matched in this file, login is automatically permitted, provided the client and server user names are the same. Additionally, successful public key authentication is typically required. This file must be writable only by a superuser. It is recommended that it be world-readable.

Guideline: Do not use user names in /etc/hosts.equiv. Be aware that the named users can log in as any user, including bin, daemon, adm, and other accounts that own critical binaries and directories. The only valid use for user names is in negative entries.

/etc/nologin

If this file exists, **sshd** refuses to let anyone except a superuser log in. The contents of the file are displayed to anyone trying to log in and non-superuser connections are refused. The file must be world-readable.

/etc/motd

Contains the message of the day. See the **sshd_config** keyword "PrintMotd" on page 155 for more information.

/etc/ssh/moduli
Contains Diffie-Hellman groups used for the Diffie-Hellman Group Exchange. The file format is described in "moduli" on page 160.

/etc/ssh/sshd_config
Contains configuration data for **sshd**. The file format and configuration options are described in "sshd_config" on page 144.

/etc/ssh/ssh_host_key, /etc/ssh/ssh_host_dsa_key, /etc/ssh/ssh_host_rsa_key
These three files contain the private parts of the host keys. They must only be owned and readable by a superuser. **sshd** does not start if these files are group-accessible or world-accessible.

/etc/ssh/ssh_host_key.pub, /etc/ssh/ssh_host_dsa_key.pub, /etc/ssh/ssh_host_rsa_key.pub
These three files contain the public parts of the host keys. These files are only provided for the convenience of the user so their contents can be copied to known hosts files. They are created using **ssh-keygen**. This file must be writable only by a superuser and can, but need not be, world-readable. Their contents must match the respective private parts.

/etc/ssh/shosts.equiv
This file is used in exactly the same way as `/etc/hosts.equiv`, but allows host-based authentication without permitting login with **rlogin** or **rsh**.

/etc/ssh/ssh_known_hosts
System-wide list of known host keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization. See "ssh_known_hosts file format" on page 122 for further details of the format of this file. This file must be writeable only by the owner and only be world-readable.

/etc/ssh/sshrc
Similar to `~/.ssh/rc`, it can be used to specify machine-specific login-time initialization globally. This file should be writable only by a superuser and world-readable.

/etc/ssh/zos_sshd_config
Contains z/OS-specific configuration data for **sshd**. The file format and configuration options are described in "zos_sshd_config" on page 158.

/var/empty
chroot directory used by **sshd** during privilege separation in the pre-authentication phase. The directory must not contain any files. It must also be owned by a superuser and not be group-writable or world-writable.

/var/run/sshd.mm.*XXXXXXXX*
Temporary files created by **sshd** for compression with privilege separation.

/var/run/sshd.pid
Contains the process ID of the **sshd** listening for connections (if there are several daemons running concurrently for different ports, this contains the process ID of the one started last). The contents of this file are not sensitive. It can be world-readable. This file is not created if the server is running in debug mode.

Environment variables

_ZOS_OPENSSH_DEBUG
Contains z/OS-specific debug information. This environment variable is only used internally and is not for external specification.

_ZOS_OPENSSH_MSGCAT
Identifies the OpenSSH message catalog to be used when sending OpenSSH error messages.

_ZOS_SMF_FD
Set to the file descriptor number used for interprocess communication during SMF-related processing. This environment variable is only used internally and is not for external specification.

_ZOS_SSHD_CONFIG
Specifies the path name of the user-defined **zos_sshd_config** configuration file. The default is `/etc/ssh/zos_sshd_config`. For a list of available keywords, see "zos_sshd_config" on page 158. The recommended permissions of the specified file are read/write for the user and not accessible by others.

Related information

moduli, **scp**, **sftp**, **sftp-server**, **ssh**, **ssh-add**, **ssh-agent**, **ssh-keygen**, **sshd_config**, **zos_sshd_config**

Authors

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0. Niels Provos and Markus Friedl contributed support for privilege separation.

Chapter 10. OpenSSH files

OpenSSH client configuration files

ssh_config — OpenSSH client configuration files

Description

ssh obtains configuration data from these sources in the following order:

1. Command line options
2. User's configuration file (`~/.ssh/config`)
3. System-wide configuration file (`/etc/ssh/ssh_config`)

For each parameter, the first obtained value is used. The **ssh_config** configuration files contain sections separated by "Host" specifications and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

Guideline: Because the first obtained value for each parameter is used, you should put host-specific declarations near the beginning of the file, and put the general defaults at the end.

File format

The **ssh_config** configuration file views empty lines and lines starting with # as comments.

Configuration options can be specified using two different formats.

- The first format is the keyword argument pair separated by white space.
- The second format is the keyword argument pair separated with exactly one "=" and optional white space. This format is useful to avoid the need to quote white space when specifying configuration options using the **scp**, **sftp** and **ssh -o** options. Arguments can optionally be enclosed in double quotes (") in order to represent arguments containing spaces.

  Example:

  ```
  keyword argument
  keyword=argument
  ```

Keywords are not case sensitive and arguments are case sensitive. Following are the possible keywords:

AddressFamily
: Specifies which address family to use when connecting. Valid arguments are "any", "inet" (for IPv4 only) or "inet6" (for IPv6 only).

AFSTokenPassing
: Not supported on z/OS UNIX. Specifies whether to pass AFS tokens to remote host. The argument to this keyword must be "yes" or "no".

  Restriction: The AFSTokenPassing option applies to protocol version 1 only.

BatchMode
: If set to "yes", passphrase/password querying is disabled. This option is

useful in scripts and other batch jobs where no user is present to supply the password. The argument must be set to "yes" or "no". The default is "no".

Rule: An SSH agent, Kerberos authentication (if available), or trusted host authentication must be used for authentication to succeed in batch mode.

BindAddress
Uses the specified address on the local machine as the source address of the connection. This option is only useful on systems with more than one address and does not work if UsePrivilegedPort is set to "yes".

ChallengeResponseAuthentication
Not supported on z/OS UNIX. Specifies whether to use challenge-response authentication. The argument must be set to "yes" or "no". The default is "yes".

CheckHostIP
If this flag is set to "yes", **ssh** checks the host IP address in the known_hosts file. Regardless of this setting, **ssh** always checks the known hosts files for the user-specified host name. Enabling this option means that both the user-specified host name and IP address should be in a known hosts file. If not, a warning is issued to inform the user that the missing entry is being written to the `~/.ssh/known_hosts` file. This flag allows **ssh** to detect if a host key changed due to DNS spoofing. If the option is set to "no", the check is not executed. The default is "yes".

Cipher
Specifies the cipher to use for encrypting the session in protocol version 1. Currently, Blowfish, Triple DES (3DES), and DES are supported. The DES cipher is only supported in the **ssh** client for interoperability with legacy protocol version 1 implementations that do not support the 3DES cipher. Its use is strongly discouraged due to cryptographic weaknesses. The default is 3DES.

Ciphers
Specifies the ciphers to use for encrypting the session in protocol version 2 in the order of preference. Multiple ciphers must be separated by commas. Valid ciphers include:

3des-cbc Triple DES algorithm (3DES)

acss@openssh.org
OpenSSH acss@openssh.org cipher

aes128-cbc Advanced Encryption Standard (AES) CBC mode with 128-bit key

aes128-ctr Advanced Encryption Standard (AES) CTR mode with 128-bit key

aes192-cbc Advanced Encryption Standard (AES) CBC mode with 192-bit key

aes192-ctr Advanced Encryption Standard (AES) CTR mode with 192-bit key

aes256-cbc Advanced Encryption Standard (AES) CBC mode with 256-bit key

aes256-ctr Advanced Encryption Standard (AES) CTR mode with 256-bit key

arcfour Arcfour algorithm

arcfour128 Arcfour algorithm with 128-bit key

arcfour256 Arcfour algorithm with 256-bit key

blowfish-cbc Blowfish algorithm

cast128-cbc CAST algorithm

rijndael-cbc@lysator.liu.se
Same as Advanced Encryption Standard (AES) CBC mode with 256-bit key

The ciphers list is typically one long unbroken line; however due to space limitations, the default ciphers list is not shown as one unbroken line. The default is:

```
aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,
3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,
rijndael-cbc@lysator.liu.se
```

Example:

```
ssh  -o"Ciphers aes128-cbc,blowfish-cbc"   Billy@us.pok.ibm.com
```

ClearAllForwardings
Specifies that all local, remote, and dynamic port forwardings specified in the configuration files or on the command line be cleared. This option is primarily useful from the **ssh** command line to clear port forwardings set in configuration files and is automatically set by **scp** and **sftp**. The argument must be set to "yes" or "no". The default is "no".

Compression
Specifies whether to use compression. The argument must be set to "yes" or "no". The default is "no".

CompressionLevel
Specifies the compression level to use if compression is enabled. The argument must be an integer from 1 (fast) to 9 (slow, best). The default level is 6, which is good for most applications.

Restriction: This option applies to protocol version 1 only.

ConnectionAttempts
Specifies the number of tries (one per second) to make before exiting. The argument must be an integer. This might be useful in scripts if the connection sometimes fails. The default is 1.

ConnectTimeout
Specifies the timeout (in seconds) used when connecting to the SSH server, instead of using the default system's TCP timeout. This value is used only when the target is down or is unreachable, not when it refuses the connection.

ControlMaster
Enables the sharing of multiple sessions over a single network connection. When set to "yes", **ssh** listens for connections on a control socket specified using the ControlPath argument. Additional sessions can connect to this socket using the same ControlPath with ControlMaster set to "no" (the default). These sessions will try to reuse the master instance's network connection rather than initiating new ones, but will fall back to connecting normally if the control socket does not exist, or is not listening.

Setting ControlMaster to "ask" causes **ssh** to listen for control connections, but requires confirmation using the SSH_ASKPASS program before they are accepted (see “ssh-add” on page 99 for details). If the ControlPath cannot be opened, **ssh** continues without connecting to a master instance.

X11 and **ssh-agent** forwarding are supported over these multiplexed connections. However, the display and agent forwarded will be the one belonging to the master connection; that is, it is not possible to forward multiple displays or agents.

Two additional options allow for opportunistic multiplexing: try to use a master connection but fall back to creating a new one if one does not exist. These options are "auto" and "autoask". The latter requires confirmation such as the "ask" option.

ControlPath
Specifies the path to the control socket used for connection sharing as described in the ControlMaster option or the string "none" to disable connection sharing. In the path, *%l* is substituted by the local host name, *%h* is substituted by the target host name, *%p* the port, and *%r* by the remote login username. To ensure that shared connections are uniquely identified, any ControlPath used for opportunistic connection sharing should include at least *%h*, *%p*, and *%r*.

Restriction: The maximum path length is 107 bytes.

DynamicForward
Specifies that a TCP port on the local machine be forwarded over the secure channel and the application protocol is then used to determine where to connect to from the remote machine. The argument must be a port number. The argument must be either *[bind_address:]port* or *[bind_address/]port*. IPv6 addresses can be specified by enclosing addresses in square brackets or by using the *[bind_address/]port* syntax. By default, the local port is bound in accordance with the GatewayPorts setting. However, an explicit bind_address can be used to bind the connection to a specific address. The bind_address of "localhost" indicates that the listening port be bound for local use only, while an empty address or '*' indicates that the port should be available from all interfaces.

Currently, the SOCKS4 and SOCKS5 protocols are supported and **ssh** will act as a SOCKS server. Multiple forwardings can be specified and additional forwarding can be given on the command line. Only the superuser can forward privileged ports.

ExitOnForwardFailure
Specifies whether **ssh** is to terminate the connection if it cannot set up all requested dynamic, tunnel, local, and remote port forwardings. The argument must be "yes" or "no". The default is "no".

EnableSSHKeysign
Setting this option to "yes" in the global client configuration file `/etc/ssh/ssh_config` enables the use of the helper program **ssh-keysign** during HostbasedAuthentication. (See “ssh-keysign” on page 114 for more information about **ssh-keysign**.) The argument must be "yes" or "no". The default is "no".

Rule: Put the EnableSSHKeysign option in the non-host-specific section.

EscapeChar
Sets the escape character (default of ~). The escape character can also be set on the command line. The argument can be a single character, ^ followed

by a letter or "none" to disable the escape character entirely (making the connection transparent for binary data).

ForwardAgent
Specifies whether the connection to the authentication agent (if any) is to be forwarded to the remote machine. The argument must be set to "yes" or "no". The default is "no".

Enable agent forwarding with caution. Users with the ability to bypass file permissions on the remote host (for the agent's UNIX-domain socket) can access the local agent through the forwarded connection. Attackers cannot obtain key material from the agent; however, they can perform operations on the keys that enable them to authenticate using the identities loaded into the agent.

ForwardX11
Specifies whether X11 connections are to be automatically redirected over the secure channel and DISPLAY set. The argument must be set to "yes" or "no". The default is "no".

Enable X11 forwarding with caution. Users with the ability to bypass file permissions on the remote host (for the user's X11 authorization database) can access the local X11 display through the forwarded connection. An attacker may then be able to perform activities such as keystroke monitoring if the ForwardX11Trusted option is also enabled.

ForwardX11Trusted
If this option is set to "yes", remote X11 clients will have full access to the original X11 display. If this option is set to "no", then remote X11 clients are considered untrusted and will be prevented from stealing or tampering with data belonging to trusted X11 clients. Furthermore, when set to "no", the xauth token (cookie) used for the session will be set to expire after 20 minutes. Remote clients will be refused access after this time. The default is "no".

See the X11 SECURITY extension specification for full details on the restrictions imposed on untrusted clients.

GatewayPorts
Specifies whether remote hosts are allowed to connect to local forwarded ports. By default, **ssh** binds local port forwardings to the loopback address. The binding prevents other remote hosts from connecting to forwarded ports. Use GatewayPorts to specify that **ssh** is to bind local port forwardings to the wildcard address, thus allowing remote hosts to connect to forwarded ports. The argument must be set to "yes" or "no". The default is "no".

GlobalKnownHostsFile
Specifies a file to use for the global host key database instead of `/etc/ssh/ssh_known_hosts`.

GSSAPIAuthentication
Not supported on z/OS UNIX. Specifies whether user authentication (such as Kerberos Authentication) based on GSS-API is allowed. The default is "no".

Restriction: The GSSAPIAuthentication option applies to protocol version 2 only.

GSS-API stands for Generic Security Services Application Programming Interface. It is a generic API for handling client-server authentication.

Because it provides security services to callers in a generic way, supportable with a range of underlying mechanisms and technologies, it allows for source-level portability of applications to different environments. For more details, check IETF standard RFC 2743 at http://www.ietf.org/rfc/rfc2743.txt.

GSSAPIDelegateCredentials
Not supported on z/OS UNIX. Forwards (delegates) credentials to the server. The default is "no".

Restriction: This option applies to protocol version 2 only.

GSS-API stands for Generic Security Services Application Programming Interface. It is a generic API for handling client-server authentication. Because it provides security services to callers in a generic way, supportable with a range of underlying mechanisms and technologies, it allows for source-level portability of applications to different environments. For more details, check IETF standard RFC 2743 at http://www.ietf.org/rfc/rfc2743.txt.

HashKnownHosts
If this option is set to "yes", indicates that **ssh** is to hash host names and addresses when they are added to `~/.ssh/known_hosts`. These hashed names can be used normally by **ssh** and **sshd**, but they do not reveal identifying information if the file's contents are disclosed. Existing names and addresses in known hosts files are not automatically converted, but can be manually hashed using **ssh-keygen**. The default is "no".

Host Restricts the following declarations (up to the next Host keyword) to be only for those hosts that match one of the patterns given after the keyword. A single `*` as a pattern can be used to provide global defaults for all hosts. The host is the hostname argument given on the command line (the name is not converted to a canonical host name before matching).

See “Patterns” on page 140 for more information about patterns.

HostbasedAuthentication
Specifies whether to try rhosts-based authentication with public key authentication. The argument must be set to "yes" or "no". The default is "no".

Restriction: This option applies to protocol version 2 only.

The HostbasedAuthentication option is similar to RhostsRSAAuthentication.

If the local host (that is, the client system) keys are only stored in a SAF key ring, then a client using host-based authentication will not be able to access those keys because it uses **ssh-keysign** which only locates host keys in the default UNIX files. However, host-based authentication for clients on the local host can still be set up by an administrator on both the local and remote hosts, as follows:

1. Generate a new public/private key pair for the local host, storing them in the default UNIX files.
2. Extract the local host's public host key from the key pair just created. Copy it into the remote host's `/etc/ssh/ssh_known_hosts` file.

HostKeyAlgorithms
Specifies the protocol version 2 ; host key algorithms that the client wants to use in order of preference. The default for this option is "ssh-rsa,ssh-dss".

HostKeyAlias
Specifies an alias that should be used instead of the real host name when looking up or saving host key in the host key database files. This option is useful for tunneling SSH connections or for multiple servers running on a single host.

HostName
Specifies the real host name to log into. You can use this option to specify nicknames or abbreviations for hosts. The default is the name given on the command line. Numeric IP addresses are also permitted both on the command line and in HostName specifications.

IdentitiesOnly
Specifies that **ssh** should only use the authentication identity files configured in the **ssh_config** files and key ring certificates configured in the **zos_user_ssh_config** file, even if the **ssh-agent** offers more identities. The argument to this keyword must be "yes" or "no". The default is "no".

Guideline: Use this option in situations where **ssh-agent** offers many different identities.

IdentityFile
Specifies a file from which the user's RSA or DSA authentication identity is read. The default is `~/.ssh/identity` for protocol version 1. For protocol version 2, the default is `~/.ssh/id_rsa` and `~/.ssh/id_dsa`. Additionally, any identities configured with the IdentityKeyRingLabel or represented by the authentication agent are used for authentication. Refer to the **-i** *identity_file* description in the **ssh** command for a summary of the order that identities are tried during public key authentication.

The file name can use the tilde syntax to refer to a user's home directory or one of the following escape characters: *%d* (local user's home directory), *%u* (local user name), *%l* (local host name), *%h* (remote host name) or *%r* (remote user name).

It is possible to have multiple identity files specified in configuration files; all these identities will be tried in sequence.

KbdInteractiveAuthentication
Not supported on z/OS UNIX. Specifies whether to use keyboard-interactive authentication. The argument to this keyword must be "yes" or "no".

KbdInteractiveDevices
Not supported on z/OS UNIX. Specifies the list of methods to use in keyboard-interactive authentication. Multiple method names must be comma-separated. The default is to use the server-specified list. The methods available vary depending on what the server supports. For an OpenSSH server, it might be zero or more instances of "bsdauth", "pam", and "skey".

KeepAlive
This keyword is supported for compatibility with versions of OpenSSH before 3.8.1p1. On systems using OpenSSH 3.8.1p1 or later, you should use the keyword TCPKeepAlive instead.

Specifies whether the system should send TCP keepalive messages to the other side. If they are sent, a lost network connection or stopping of one of the machines will be properly noticed. However, this means that OpenSSH connections will end if the route is down temporarily.

The default is "yes" (to send keepalives), and the client will notice if the network goes down or the remote host dies. This is important in scripts as well as to many users. To disable keepalives, set the value to "no".

LocalCommand
Specifies a command to be executed on the local machine after successfully connecting to the server. The command string extends to the end of the line, and is executed with the user's shell. This option is ignored unless PermitLocalCommand has been enabled.

LocalForward
Specifies that a TCP port on the local machine is to be forwarded over the secure channel to the specified host and port from the remote machine. The first argument must be *[bind_address:]port* and the second must be *host:hostport*. IPv6 addresses can be specified by enclosing addresses in square brackets or by using an alternate syntax: *[bind_address/]port* and *host/hostport*. Multiple forwardings can be specified and additional forwardings can be given on the command line. Only the superuser can forward privileged ports. By default, the local port is bound in accordance with the GatewayPorts setting. However, an explicit *bind_address* can be used to bind the connection to a specific address. The *bind_address* of "localhost" indicates that the listening port is to be bound for local use only, while an empty address or '*' indicates that the port is to be available from all interfaces.

LogLevel
Gives the verbosity level that is used when logging messages from **ssh**. The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. The default is INFO. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of verbose output.

MACs Specifies the MAC (message authentication code) algorithms in order of preference. The MAC algorithm is used for data integrity protection. Multiple algorithms must be comma-separated.

The MAC algorithms list is typically one long unbroken line; however due to space limitations, the default MAC algorithms list is not shown as one unbroken line. The default is: `hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96`.

Restriction: This option applies to protocol version 2 only.

NoHostauthenticationForLocalhost
This option can be used if the home directory is shared across machines (for example, if the home directory is NFS-mounted to multiple machines). In this case, localhost will refer to a different machine on each of the machines and the user will get many warnings about changed host keys. However, this option disables host authentication for localhost (to avoid these warnings). The argument must be set to "yes" or "no". The default is to check the host key for localhost.

NumberOfPasswordPrompts
Specifies the number of password prompts before giving up. The argument must be an integer. The default is 3.

Regardless of this value, the SSH daemon still regulates the total number of authentication attempts.

PasswordAuthentication
Specifies whether to use password authentication. The argument must be set to "yes" (default) or "no". Password authentication prompts the user for a password or password phrase that is sent to the remote host for checking.

PermitLocalCommand
Allows local command execution by means of the LocalCommand option or using the !command escape sequence in **ssh**. The argument must be "yes" or "no". The default is "no".

Port Specifies the port number to connect to on the remote host. The default is 22.

PreferredAuthentications
Specifies the order in which the client should try protocol version 2 authentication methods. This allows a client to prefer one method (such as publickey) over another method (such as password). The default for this option is *hostbased,publickey,keyboard-interactive,password*.

gssapi-with-mic and *keyboard-interactive* are not supported on z/OS UNIX.

Protocol
Specifies the protocol versions **ssh** should support in order of preference. The possible values are 1 and 2. Multiple versions must be comma-separated. The default is 2. If 2,1 is specified, **ssh** tries version 2 and falls back to version 1 if version 2 is not available.

ProxyCommand
Specifies the command to use to connect to the server. The command string extends to the end of the line and is executed with the user's shell. In the command string, *%h* will be substituted by the host name to connect and *%p* by the port. The command can be basically anything and should read from its standard input and write to its standard output. It should eventually connect an **sshd** server running on some machine or execute **sshd –i**. Host key management will be done using the HostName of the host being connected (defaulting to the name typed by the user). The CheckHostIP keyword is not available for connects with a proxy command.

PubkeyAuthentication
Specifies whether to try public key authentication for protocol version 2. The argument must be set to "yes" (default) or "no".

RekeyLimit
Specifies the maximum amount of data that can be transmitted before the session key is renegotiated. The argument is the number of bytes, with an optional suffix of K, M, or G to indicate kilobytes, megabytes, or gigabytes, respectively. The default is between 1G and 4G, depending on the cipher.

Restrictions:
- This option applies to protocol version 2 only.
- The maximum value is UINT_MAX bytes and the minimum value is 16 bytes.

RemoteForward
Specifies that a TCP port on the remote machine is to be forwarded over the secure channel to the specified host and port from the local machine. The argument must be either *[bind_address:]port* or *[bind_address/]port*, and the second must be *host :hostport*. IPv6 addresses can be specified by enclosing addresses in square brackets or by using the *[bind_address/]port*

syntax for the first argument and *host/hostport* in the second argument. Multiple forwardings can be specified and additional forwardings can be given on the command line.

If the *bind_address* is not specified, the default is to only bind to loopback addresses. If the *bind_address* is '*' or an empty string, then the forwarding is requested to listen on all interfaces. Specifying a remote *bind_address* succeeds only if the server's GatewayPorts option is enabled as described in “GatewayPorts” on page 133.

Restriction: Only the superuser can forward privileged ports.

RhostsAuthentication
Specifies whether to try rhosts-based authentication in protocol version 1. This declaration only affects the client side and does not affect security. Most servers do not permit RhostsAuthentication because it is not secure. The argument must be set to "yes" or "no". The default is "no".

Requirement: ssh must be setuid 0 and UsePrivilegedPort must be set to "yes".

When connecting to **sshd** running on a non-z/OS platform using this option, this form of authentication might fail if the server side of OpenSSH version is 3.7 or higher, because RhostsAuthentication is no longer supported at these levels.

Restriction: RhostsAuthentication cannot be used with privilege separation.

RhostsRSAAuthentication
Specifies whether to try rhosts-based authentication with RSA host authentication in protocol version 1. This option requires **ssh** to be setuid 0. The argument must be set to "yes" or "no". The default is "no".

RSAAuthentication
Specifies whether to try RSA authentication. The argument to this keyword must be "yes" (default) or "no". RSA authentication will only be attempted if the identity file exists, or an authentication agent is running.

Restriction: This option applies to protocol version 1 only.

SendEnv
Specifies which environment variables from the local environment variables are to be sent to the server. Environment variables are specified by name, which can contain wildcard characters. However, the name cannot contain the equal (=) character. Multiple environment variables can be separated by white space or spread across multiple SendEnv options for a maximum of 256 environment variable specifications. The default is not to send any environment variables.

See “Patterns” on page 140 for more information about patterns.

The accepted environment variables are processed after authentication but before general environment variable setup and handling of the **sshd_config** keyword PermitUserEnvironment. Therefore, the values of accepted environment variables might be overwritten as a result of this subsequent processing.

Restriction: Environment variable passing is only supported in protocol version 2. The server must also support environment variable passing and the server must be configured to accept these environment variables. See

the description of the **sshd_config** keyword “AcceptEnv” on page 144 for information about configuring the server.

ServerAliveInterval
Sets a timeout interval in seconds after which if no data has been received from the server, **ssh** sends a message through the encrypted channel to request a response from the server. The default is 0, indicating that these messages are not sent to the server.

Restriction: This option applies to protocol version 2 only.

ServerAliveCountMax
Sets the number of server alive messages that can be sent without **ssh** receiving any messages back from the server. If this threshold is reached while server alive messages are being sent, **ssh** disconnects from the server, thus ending the session. The default value is 3.

Example: If ServerAliveInterval is set to 15, and ServerAliveCountMax is left at the default, if the server becomes unresponsive **ssh** will disconnect after approximately 45 seconds.

Note: The use of server alive messages is very different from TCPKeepAlive. The server alive messages are sent through the encrypted channel and therefore are not spoofable. The TCP keepalive option enabled by TCPKeepAlive is spoofable. The server alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

Restriction: This option applies to protocol version 2 only.

SmartcardDevice
Not supported on z/OS UNIX. Specifies which smart card device to use. The argument to this keyword is the device that **ssh** should use to communicate with a smart card used for storing the user's private RSA key. By default, no device is specified and smart card support is not activated.

StrictHostKeyChecking
If the argument is set to "yes", **ssh** will never automatically add host keys to the `~/.ssh/known_hosts` file and will refuse to connect to a host whose host key has changed. This provides maximum protection against trojan horse attacks, but can be troublesome when the `/etc/ssh/ssh_known_hosts` file is poorly maintained or connections to new hosts are frequently made. This option forces the user to manually add all new hosts. If the argument is set to "no", **ssh** will automatically add new host keys to the user known hosts files. If the flag is set to "ask", new host keys will be added to the user known host files only after the user has confirmed the action and **ssh** will refuse to connect to hosts whose host key has changed. The host keys of known hosts will be verified automatically in all cases. The argument must be set to "yes", "no", or "ask". The default is "ask".

TCPKeepAlive
Specifies whether the system should send TCP keepalive messages to the other side. If they are sent, a lost network connection or stopping of one of the machines will be properly noticed. However, this means that OpenSSH connections will end if the route is down temporarily. The default is "yes" (to send TCP keepalive messages), and the client will notice if the network goes down or the remote host dies. This is important in scripts as well as to many users. To disable TCP keepalive messages, set the value to "no".

Tunnel
Not supported on z/OS UNIX. Requests tunnel device forwarding between the client and the server. The argument must be "yes", "point-to-point" (layer 3), "ethernet" (layer 2), or "no". Specifying "yes" requests the default tunnel mode, which is "point-to-point". The default is "no".

TunnelDevice
Not supported on z/OS UNIX. Specifies the tunnel devices to open on the client (*local_tun*) and the server (*remote_tun*).

The argument must be *local_tun*[:*remote_tun*]. The devices can be specified by numerical ID or the keyword "any", which uses the next available tunnel device. If *remote_tun* is not specified, it defaults to "any". The default is "any:any".

UsePrivilegedPort
Specifies whether to use a privileged port for outgoing connections. The argument must be set to "yes" or "no". The default is "no".

Rules:

- This option must be set to "yes" if RhostsAuthentication and RhostsRSAAuthentication authentications are needed with servers that only support protocol version 1.
- If UsePrivilegedPort is set to "yes", **ssh** must be setuid 0.

User Specifies the name that the user can use when logging on. This can be useful when a different user name is used on different machines. You do not have to remember to give the user name on the command line.

UserKnownHostsFile
Specifies a file to use for the user host key database instead of `~/.ssh/known_hosts`.

VerifyHostKeyDNS
Specifies whether to verify the remote key using DNS and SSHFP (SSH fingerprint) resource records. If this option is set to "yes", the client will implicitly trust keys that match a secure fingerprint from DNS. Insecure fingerprints are handled as if this option was set to "ask". If this option is set to "ask", information about fingerprint match is displayed, but the user will still need to confirm new host keys according to the StrictHostKeyChecking option. The argument must be "yes", "no" or "ask". The default is "no".

Restriction: This option applies to protocol version 2 only.

XAuthLocation
Specifies the full path name of the xauth program. The default is `/usr/X11R6/bin/xauth`. For more information, see “Steps for configuring the system for X11 forwarding” on page 47.

Patterns

A *pattern* consists of zero or more non-white space characters, '*' (a wildcard that matches zero or more characters), or '?' (a wildcard that matches exactly one character). For example, to specify a set of declarations for any host in the ".co.uk" set of domains, the following pattern could be used:

```
Host *.co.uk
```

The following pattern would match any host in the 192.168.0.[0-9] network range:

```
Host 192.168.0.?
```

A *pattern-list* is a comma-separated list of patterns. Patterns within pattern-lists can be negated by preceding them with an exclamation mark ('!'). For example, to allow a key to be used from anywhere within an organization except from the "dialup" pool, the following entry (in the `authorized_keys` file) could be used:

```
from="!*.dialup.example.com,*.example.com"
```

Limitations

Due to limitations in the SECSH protocol with regards to EBCDIC platforms, user-defined subsystems are only supported between z/OS and z/OS. (For information about the IETF SECSH RFCs and internet drafts, see Appendix C, “RFCs and Internet drafts,” on page 339.)

Files

~/.ssh/config
: The per-user configuration file. For the format of this file, see “File format” on page 129. The file is used by the SSH client. Because of the potential for abuse, this file must have strict permissions: read/write for the user, and not writeable by others.

/etc/ssh/ssh_config
: The system-wide configuration file. This file provides defaults for those values that are not specified in the user's configuration file and for those users who do not have a configuration file. This file must be world-readable.

Related information

scp, **sftp**, **ssh**

Authors

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

zos_ssh_config — z/OS-specific system-wide OpenSSH client configuration file

Description

z/OS obtains z/OS-specific system-wide OpenSSH client configuration data only from the `/etc/ssh/zos_ssh_config` configuration file. It contains sections separated by "Host" specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

Restriction: z/OS-specific keywords cannot be specified in the **ssh_config** configuration files, such as the system-wide configuration file (`/etc/ssh/ssh_config`) or user-defined configuration file specified with the **ssh -F** option.

File format

The **zos_ssh_config** configuration file views empty lines and lines starting with # as comments. Configuration options can be specified using two different formats.

- The first format is the keyword argument pair separated by white space.
- The second format is the keyword argument pair separated with exactly one "=" and optional white space. Arguments can optionally be enclosed in double quotes (") in order to represent arguments containing spaces.

Example:

```
keyword argument
keyword=argument
```

Keywords are not case sensitive and arguments are case sensitive. Following are the possible keywords:

ClientSMF
Specifies whether to collect client SMF records. The argument must be set to "TYPE119_U83", "TYPE119_U84" or "none". The default is "none". If set to "TYPE119_U83" or "TYPE119_U84" , SMF Type 119 client transfer completion records (subtype 97) are collected for the **sftp** and **scp** commands. SMF record exit IEFU83 receives control for "TYPE119_U83". SMF record exit IEFU84 receives control for "TYPE119_U84".

Restriction: Because this keyword can only be set in the z/OS-specific system-wide configuration file (`/etc/ssh/zos_ssh_config`), it cannot be specified using the **-o** option of **scp**, **sftp** or **ssh**.

The IEFU83 and IEFU84 exits are documented in *z/OS MVS Installation Exits*.

Host Restricts the following declarations (up to the next Host keyword) to be only for those hosts that match one of the patterns given after the keyword. A single * as a pattern can be used to provide global defaults for all hosts. The host is the hostname argument given on the command line (the name is not converted to a canonical host name before matching).

See “Patterns” on page 140 in **ssh_config** for more information about patterns.

Files

/etc/ssh/zos_ssh_config
z/OS-specific system-wide client configuration file. This file must be world-readable but writable only by a superuser.

Related information

scp, **sftp**, **ssh**

zos_user_ssh_config — z/OS-specific per-user OpenSSH client configuration file

Description

z/OS obtains z/OS-specific per-user client configuration data in the following order:

1. User-specific client options from:
   a. The command-line specification using the **-o** option of the **scp**, **sftp**, or **ssh** command.
   b. The file specified with variable _ZOS_USER_SSH_CONFIG. The default is `~/.ssh/zos_user_ssh_config`.
2. System-wide client options from the file `/etc/ssh/zos_ssh_config`.

For each keyword that only supports one instance, the first obtained value is used. If the keyword supports multiple instances, all values are obtained from all sources and used as defined by the keyword.

Restriction: z/OS-specific keywords cannot be specified in the **ssh_config** configuration files, such as the system-wide configuration file (`/etc/ssh/ssh_config`) or user-defined configuration file specified with the **ssh -F** option.

The configuration file contains sections separated by "Host" specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

File format

The **zos_user_ssh_config** configuration file views empty lines and lines starting with # as comments. Configuration options can be specified using two different formats.

- The first format is the keyword argument pair separated by white space.
- The second format is the keyword argument pair separated with exactly one "=" and optional white space. This format is useful to avoid the need to quote white space when specifying configuration options using the **scp**, **sftp** and **ssh -o** options. Arguments can optionally be enclosed in double quotes (") in order to represent arguments containing spaces.

  Example:

  ```
  keyword argument
  keyword=argument
  ```

Keywords are not case sensitive and arguments are case sensitive. Following are the possible keywords:

Host Restricts the following declarations (up to the next Host keyword) to be only for those hosts that match one of the patterns given after the keyword. A single * as a pattern can be used to provide global defaults for all hosts. The host is the hostname argument given on the command line (the name is not converted to a canonical host name before matching).

See “Patterns” on page 140 in **ssh_config** for more information about patterns.

IdentityKeyRingLabel

Specifies the key ring owner, key ring name and certificate label within the key ring from which the user's RSA or DSA authentication identity is read. The key ring can be real or virtual, and the certificate label can contain embedded blanks. The key ring and the certificate connected to the key ring were created in the user authentication setup, which is described in “Steps for setting up user authentication when keys are stored in UNIX files” on page 66. One or more blanks separate the key ring name from the certificate label. The user's RSA or DSA authentication identity is read from all certificates before the identities associated with files specified with IdentityFile are checked. Refer to the **-i** *identity_file* description in “ssh” on page 85 for a summary of the order that identities are tried during public key authentication.

The default is to use only the identity files and agent.

It is possible to have multiple identity files and key ring certificates in configuration files. If both identity files and key ring certificates are used, the key ring certificates are tried first. The maximum combined number of identity key files and key ring certificates that can be specified is 100.

The option value must be surrounded with double quotes.

Example: An example of this option in the **zos_user_ssh_config** file for a key ring named 'SSHring' that is owned by 'KeyRingOwnerID' and a certificate labeled 'my label with blanks' is as follows:

```
IdentityKeyRingLabel="KeyRingOwnerID/SSHring my label with blanks"
```

If the option is specified as a command-line option, you might need to escape the double quote characters that surround the argument value:

```
-o IdentityKeyRingLabel="\"KeyRingOwnerID/SSHring my label with blanks\""
```

Environment variable

_ZOS_USER_SSH_CONFIG
Specifies the path name of the z/OS-specific per-user OpenSSH client configuration file. The system-wide default is `/etc/ssh/zos_ssh_config` and the user's default is `~/.ssh/zos_user_ssh_config`. If this variable is specified, it replaces the user's default file but not the system-wide default file. The recommended permissions of the specified file are read/write for the user and not accessible by others.

Files

~/.ssh/zos_user_ssh_config
z/OS-specific per-user OpenSSH client configuration file. This file must be writable only by the user. It can be readable by others, but need not be.

Related information

scp, **sftp**, **ssh**

OpenSSH daemon configuration files

sshd_config — OpenSSH daemon configuration file

Description

sshd reads configuration data from the `/etc/ssh/sshd_config` file or the file specified with **-f** on the command line. “File format” describes the file format.

File format

The **sshd_config** configuration file views empty lines and lines starting with # as comments.

Configuration options can be specified using two different formats.

- The first format is the keyword argument pair separated by white space.
- The second format is the keyword argument pair separated with exactly one "=" and optional white space. This format is useful to avoid the need to quote white space when specifying configuration options using the **sshd –o** options. Arguments can optionally be enclosed in double quotes (") in order to represent arguments containing spaces.

  Example:

  ```
  keyword argument
  keyword=argument
  ```

Keywords are not case sensitive and arguments are case sensitive. Following are possible keywords:

AcceptEnv
Specifies which environment variables sent by the client will be copied into the session's environment. See the description of the **ssh_config** keyword

"SendEnv" on page 138 for information about configuring clients. Variables are specified by name, which can contain the wildcard characters '*' and '?'. However, the name cannot contain the equal (=) character. Multiple environment variables can be separated by white spaces or spread across multiple AcceptEnv options for a maximum of 256 environment variable specifications. The default is not to accept any environment variables.

Guideline: Be careful when using the AcceptEnv option because some environment variables can be used to bypass restricted user environments.

The accepted environment variables are processed after authentication but before general environment variable setup and handling of the **sshd_config** keyword PermitUserEnvironment. Therefore, the values of accepted environment variables might be overwritten as a result of this subsequent processing.

Restriction: Environment variable passing is supported for protocol version 2 only.

AddressFamily
Specifies the address family to be used by **sshd**. Valid arguments are "any", "inet" (use IPv4 only), or "inet6" (use IPv6 only). The default is "any".

AFSTokenPassing
Not supported on z/OS UNIX. Specifies whether an AFS token can be forwarded to the server. The default is "no".

AllowGroups
This keyword can be followed by a list of group name patterns, separated by spaces. If specified, login is allowed only for users whose primary group or supplementary group list matches one of the patterns. Only group names are valid; a numerical group ID is not recognized. By default, login is allowed for all groups. The allow and deny options are processed in the following order to determine if the user should be disallowed from login: DenyUsers, AllowUsers, DenyGroups, and then AllowGroups. To be allowed to login, you must pass all the tests for the specified keywords.

That is, if you want *userx* who is in *groupy* and *groupz* to be allowed to login, and you plan to specify all four keywords, then:

- *userx* must not be in DenyUsers, and
- *userx* must be in AllowUsers, and
- both *groupy* and *groupz* must not be in DenyGroups, and
- either *groupy* or *groupz* must be in AllowGroups

Note: To be allowed to login, the user must have a group if AllowGroups or DenyGroups is specified.

See "Patterns" on page 140 in **ssh_config** for more information about patterns.

Refer to the **sshd_config** keyword "Match" on page 152 for more information about matching z/OS user and group names.

Restriction: The maximum number of AllowGroups specifications is 256.

AllowTcpForwarding
Specifies whether TCP forwarding is permitted. Disabling TCP forwarding does not improve general z/OS security unless users are also denied shell access, because they can install their own forwarders. The default is "no".

AllowUsers

This keyword can be followed by a list of user name patterns, separated by spaces. If specified, login is allowed only for user names that match one of the patterns. Only user names are valid; a numerical user ID is not recognized. If the pattern takes the form *user@host*, then *user* and *host* are separately checked, restricting logins to particular users from particular hosts. The default is to allow login for all users. The allow and deny options are processed in the following order to determine if the user should be disallowed from login: DenyUsers, AllowUsers, DenyGroups, and then AllowGroups. To be allowed to login, you must pass all the tests for the specified keywords.

That is, if you want *userx* who is in *groupy* and *groupz* to be allowed to login, and you plan to specify all four keywords, then:

- *userx* must not be in DenyUsers, and
- *userx* must be in AllowUsers, and
- both *groupy* and *groupz* must not be in DenyGroups, and
- either *groupy* or *groupz* must be in AllowGroups

Note: To be allowed to login, the user must have a group if AllowGroups or DenyGroups is specified.

See “Patterns” on page 140 in **ssh_config** for more information about patterns.

Refer to the **sshd_config** keyword “Match” on page 152 for more information about matching z/OS user and group names.

Restriction: The maximum number of AllowUsers specifications is 256.

AuthorizedKeysFile

Specifies the file that contains the public keys that can be used for user authentication. AuthorizedKeysFile can contain tokens in the form *%T* which are substituted during connection setup. The following tokens are defined: %% is replaced by a literal %, *%h* is replaced by the home directory of the user being authenticated and *%u* is replaced by the username of that user. After expansion, AuthorizedKeysFile is taken to be an absolute path or one relative to the user's home directory (if no absolute path is given). The default is `.ssh/authorized_keys` anchored off the user's home directory.

Restriction: The maximum path length is 1023 bytes.

Banner

The contents of the specified file are sent to the remote user before authentication is allowed. If the argument is "none", then no banner is displayed. The default is no banner is displayed.

Restriction: This option applies to protocol version 2 only.

ChallengeResponseAuthentication

Not supported on z/OS UNIX. Specifies whether challenge-response authentication is allowed. The default is "no".

ChrootDirectory

Specifies a path to chroot to after authentication. This path, and all its components, must be root-owned directories that are not writable by any other user or group. This path also affects the files used during the login process. The default is not to chroot. For more information, see “Login process” on page 119 in the **sshd** section.

The path can contain the following tokens that are expanded at runtime once the connecting user has been authenticated: %% is replaced by a literal %, %*h* is replaced by the home directory of the user being authenticated, and %*u* is replaced by the username of that user.

The ChrootDirectory must contain the necessary files and directories to support the users' session. For interactive sessions, a shell (typically, sh) is required as well as basic /dev nodes such as null, zero, stdin, stdout, stderr, random and tty devices. For file transfer sessions using **sftp,** no additional configuration of the environment is necessary if the in-process **sftp** server is used (see "Subsystem" on page 156 for details).

Rule: If the syslog daemon (**syslogd**) is used to debug the users' session, such as a file transfer session using **sftp**, then the ChrootDirectory must contain the datagram socket in use by **syslogd** (for example, `/dev/log`).

Restriction: The maximum path length is 1023 bytes.

Ciphers

Specifies the ciphers to use for encrypting the session in protocol version 2. Multiple ciphers must be comma-separated. Valid ciphers include:

3des-cbc Triple-DES (3DES) algorithm

acss@openssh.org
OpenSSH acss@openssh.org cipher

aes128-cbc Advanced Encryption Standard (AES) CBC mode with 128-bit key

aes128-ctr Advanced Encryption Standard (AES) CTR mode with 128-bit key

aes192-cbc Advanced Encryption Standard (AES) CBC mode with 192-bit key

aes192-ctr Advanced Encryption Standard (AES) CTR mode with 192-bit key

aes256-cbc Advanced Encryption Standard (AES) CBC mode with 256-bit key

aes256-ctr Advanced Encryption Standard (AES) CTR mode with 256-bit key

arcfour Arcfour algorithm

arcfour128 Arcfour algorithm with 128-bit key

arcfour256 Arcfour algorithm with 256-bit key

blowfish-cbc Blowfish algorithm

cast128-cbc CAST algorithm

rijndael-cbc@lysator.liu.se
Same as Advanced Encryption Standard (AES) CBC mode with 256-bit key

The ciphers list is typically one long unbroken line; however due to space limitations, the default ciphers list is not shown as one unbroken line. The default is:

```
aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,
3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,
rijndael-cbc@lysator.liu.se
```

ClientAliveInterval
Sets a timeout interval in seconds after which if no data has been received from the client, **sshd** sends a message through the encrypted channel to request a response from the client. The default is 0, indicating that these messages will not be sent to the client.

Restriction: This option applies to protocol version 2 only.

ClientAliveCountMax
Sets the number of client alive messages that can be sent without **sshd** receiving any messages back from the client. If this threshold is reached while client alive messages are being sent, **sshd** disconnects the client, thus terminating the session. It is important to note that the use of client alive messages is very different from TCPKeepAlive. Because the client alive messages are sent through the encrypted channel, they will not be spoofable. The TCP keepalive option enabled by TCPKeepAlive is spoofable. The client alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

If ClientAliveInterval is set to 15 and ClientAliveCountMax is left at the default value of 3, unresponsive SSH clients are disconnected after approximately 45 seconds.

Restriction: This option applies to protocol version 2 only.

Compression
Specifies whether compression is allowed (full) or delayed until the user has authenticated successfully. The argument must be set to one of the following:

- "no" to disable all compression.
- "yes" to enable both full (zlib) and delayed (zlib@openssh.com) compression.
- "delayed" to enable delayed (zlib@openssh.com) compression only.

The default is "no".

If you use compression with privilege separation, make sure that the **sshd** daemon address space can memory map at least 656 pages. Either specify at least MAXMMAPAREA (656) in BPXPRMxx to provide a large enough system-wide value, or use a security product such as RACF to specify the MMAPAREAMAX limit for the user ID starting the **sshd** daemon. See *z/OS UNIX System Services Planning* for more information about MMAPAREAMAX.

DenyGroups
This keyword can be followed by a list of group name patterns, separated by spaces. Login is disallowed for users whose primary group or supplementary group list matches one of the patterns. Only group names are valid; a numerical group ID is not recognized. The default is to allow login for all groups. The allow and deny options are processed in the following order to determine if the user should be disallowed from login: DenyUsers, AllowUsers, DenyGroups, and then AllowGroups. To be allowed to login, you must pass all the tests for the specified keywords.

That is, if you want *userx* who is in *groupy* and *groupz* to be allowed to login, and you plan to specify all four keywords, then:

- *userx* must not be in DenyUsers, and
- *userx* must be in AllowUsers, and
- both *groupy* and *groupz* must not be in DenyGroups, and

- either *groupy* or *groupz* must be in AllowGroups

Note: To be allowed to login, the user must have a group if AllowGroups or DenyGroups is specified.

See "Patterns" on page 140 in **ssh_config** for more information about patterns.

Refer to the **sshd_config** keyword "Match" on page 152 for more information about matching z/OS user and group names.

Restriction: The maximum number of DenyGroups specifications is 256.

DenyUsers

This keyword can be followed by a list of user name patterns, separated by spaces. Login is disallowed for user names that match one of the patterns. Only user names are valid; a numerical user ID is not recognized. The default is to allow login for all users. If the pattern takes the form *user@host* then *user* and *host* are separately checked, restricting logins to particular users from particular hosts. The allow and deny options are processed in the following order to determine if the user should be disallowed from login: DenyUsers, AllowUsers, DenyGroups, and then AllowGroups. To be allowed to login, you must pass all the tests for the specified keywords.

That is, if you want *userx* who is in *groupy* and *groupz* to be allowed to login, and you plan to specify all four keywords, then:

- *userx* must not be in DenyUsers, and
- *userx* must be in AllowUsers, and
- both *groupy* and *groupz* must not be in DenyGroups, and
- either *groupy* or *groupz* must be in AllowGroups

Note: To be allowed to login, the user must have a group if AllowGroups or DenyGroups is specified.

See "Patterns" on page 140 in **ssh_config** for more information about patterns.

Refer to the **sshd_config** keyword "Match" on page 152 for more information about matching z/OS user and group names.

Restriction: The maximum number of DenyUsers specifications is 256.

ForceCommand

Forces the execution of the command specified by ForceCommand, ignoring any command supplied by the client and `~/.ssh/rc` if present. The command is invoked by using the user's login shell with the **-c** option. This applies to shell, command, or subsystem execution. It is most useful inside a Match block. The command originally supplied by the client is available in the SSH_ORIGINAL_COMMAND environment variable.

Specifying a command of "internal-sftp" forces the use of an in-process sftp server that requires no support files when used with ChrootDirectory.

Tip: sftp-server options can be specified with the "internal-sftp" command by separating the options with blank spaces.

GatewayPorts

Specifies whether remote hosts are allowed to connect to ports forwarded by the client. By default, **sshd** binds remote port forwardings to the loopback address. This prevents other remote hosts from connecting to

forwarded ports. GatewayPorts can be used to specify that **sshd** is to allow remote port forwardings to bind to non-loopback addresses, thus allowing other hosts to connect. The argument can be set to one of the following:

- "no" to force remote port forwardings to be available to the local host only.
- "yes" to force remote port forwardings to bind to the wildcard address.
- "clientspecified" to allow the client to select the address to which the forwarding is bound.

The default is "no".

GSSAPIAuthentication
Not supported on z/OS UNIX. Specifies whether user authentication based on GSS-API is allowed. The default is "no".

Restriction: This option applies to protocol version 2 only.

GSS-API stands for Generic Security Services Application Programming Interface. It is a generic API for handling client-server authentication. Because it provides security services to callers in a generic way, supportable with a range of underlying mechanisms and technologies, it allows for source-level portability of applications to different environments. For more details, check IETF standard RFC 2743 at http://www.ietf.org/rfc/rfc2743.txt.

GSSAPICleanupCredentials
Not supported on z/OS UNIX. Specifies whether to automatically clear the user's credentials cache on logout. The default is "yes".

Restriction: This option applies to protocol version 2 only.

GSS-API stands for Generic Security Services Application Programming Interface. It is a generic API for handling client-server authentication. Because it provides security services to callers in a generic way, supportable with a range of underlying mechanisms and technologies, it allows for source-level portability of applications to different environments. For more details, check IETF standard RFC 2743 at http://www.ietf.org/rfc/rfc2743.txt.

HostbasedAuthentication
Specifies whether rhosts or `/etc/hosts.equiv` authentication together with successful public key client host authentication is allowed (host-based authentication). The default is "no".

Restriction: This option applies to protocol version 2 only and is similar to RhostsRSAAuthentication.

HostbasedUsesNameFromPacketOnly
Specifies whether or not the server will attempt to perform a reverse name lookup when matching the name in the `~/.shosts`, `~/.rhosts`, and `/etc/hosts.equiv` files during HostbasedAuthentication. A setting of "yes" means that **sshd** uses the name supplied by the client instead of attempting to resolve the name from the TCP connection itself. The default is "no".

HostKey
Specifies a file containing a private host key used by OpenSSH. The default host key is `/etc/ssh/ssh_host_key` for protocol version 1. For protocol version 2, the default host key is `/etc/ssh/ssh_host_rsa_key` and `/etc/ssh/ssh_host_dsa_key`. **sshd** will refuse to use a file if it is

group/world-accessible. RSA1 keys are used for protocol version 1 and DSA or RSA are used for protocol version 2.

It is possible to have multiple host key files and key ring certificates (as configured by the HostKeyRingLabel option in the **zos_sshd_config** file) in configuration files. If both host key files and key ring certificates are listed, the key ring certificates will be tried first. Only the first key found of each key type (for example, RSA, DSA, or RSA1) is used.

The maximum combined number of host key files and key ring certificates that can be specified is 256.

IgnoreRhosts
Specifies that `.rhosts` and `.shosts` files will not be used in RhostsAuthentication, RhostsRSAAuthentication or HostbasedAuthentication.

The `/etc/hosts.equiv` and `/etc/ssh/shosts.equiv` files are still used. The default is "yes".

IgnoreUserKnownHosts
Specifies whether **sshd** should ignore the user's `~/.ssh/known_hosts` during RhostsRSAAuthentication or HostbasedAuthentication. The default is "no".

KbdInteractiveAuthentication
Not supported on z/OS UNIX. Specifies whether to use keyboard-interactive authentication. The argument to this keyword must be "yes" or "no".

KeepAlive
This keyword is supported for compatibility with versions of OpenSSH before 3.8.1p1. On systems using OpenSSH 3.8.1p1 or later, you should use the keyword TCPKeepAlive instead.

Specifies whether the system should send TCP keepalive messages to the other side. If they are sent, death of the connection or crash of one of the machines will be properly noticed. However, connections will die if the route is down temporarily. On the other hand, if keepalives are not sent, sessions may hang indefinitely on the server, leaving ghost users and consuming server resources.

The default is "yes" (to send keepalives), and the server will notice if the network goes down or the client host crashes. This avoids infinitely hanging sessions.

To disable keepalives, the value should be set to "no".

KerberosAuthentication
Not supported on z/OS UNIX. Specifies whether Kerberos authentication is allowed. The authentication can be in the form of a Kerberos ticket, or if PasswordAuthentication is "yes", the password provided by the user will be validated through the Kerberos KDC. To use this option, the server needs a Kerberos servtab which allows the verification of the KDC's identity. The default is "no".

KerberosGetAFSToken
Not supported on z/OS UNIX. If AFS is active and the user has a Kerberos 5 TGT, attempts to acquire an AFS token before accessing the user's home directory. The default is "no".

KerberosOrLocalPasswd
Not supported on z/OS UNIX. Validates the password by means of the

security product's normal password checking if password authentication through Kerberos fails. The default is "yes".

KerberosTgtPassing
Not supported on z/OS UNIX. Specifies whether a Kerberos TGT is to be forwarded to the server. This will work only if the Kerberos server is actually an AFS kaserver. The default is "no".

KerberosTicketCleanup
Not supported on z/OS UNIX. Specifies whether to automatically erase the user's ticket cache file on logout. The default is "yes".

KeyRegenerationInterval
In protocol version 1, the ephemeral server key is automatically regenerated after this many seconds (if it has been used). Regeneration prevents the decrypting of captured sessions by later breaking into the machine and stealing the keys. The key is never stored anywhere. If the value is 0, the key is never regenerated. The default is 3600 (seconds).

ListenAddress
Specifies the local addresses **sshd** should listen on. The following forms can be used:

```
ListenAddress host|IPv4addr|IPv6_addr
ListenAddress host|IPv4_addr:port
ListenAddress [host|IPv6_addr]:port
```

If port is not specified, **sshd** listens on the address and all prior Port options specified. Multiple ListenAddress options are permitted. Additionally, any Port options must precede this option for non-port qualified addresses. The default is to listen on all local addresses.

LoginGraceTime
The server disconnects after this time if the user has not successfully logged in. If the value is 0, there is no time limit. The default is 120 (seconds).

LogLevel
Gives the verbosity level that is used when logging messages from **sshd**. The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. The default is INFO. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of debugging output.

Guideline: Do not log with a DEBUG level because doing so violates the privacy of users.

For more information about these logging levels, also referred to as priority codes, see the syslog daemon chapter in *z/OS Communications Server: IP Configuration Reference*.

MACs Specifies the MAC (message authentication code) algorithms in order of preference. The MAC algorithm is used for data integrity protection. Multiple algorithms must be comma-separated.

The MAC algorithms list is typically one long unbroken line; however due to space limitations, the default MAC algorithms list is not shown as one unbroken line. The default is: `hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96`.

Restriction: This option applies to protocol version 2 only.

Match Introduces a conditional block. If all of the criteria on the Match line are satisfied, the keywords on the following lines override those set in the global section of the config file, until either another Match line or the end of the file.

Rule: Global settings must be placed before the first Match block.

The arguments to Match are one or more criteria-pattern pairs. The available criteria are User, Group, Host, and Address. The match patterns can consist of single entries or comma-separated lists and can use the wildcard and negation operators described in the **ssh_config** section "Patterns" on page 140.

Restrictions:

- Only a subset of keywords can be used on the lines following a Match keyword. Those keywords are AllowTcpForwarding, Banner, ChrootDirectory, ForceCommand, GatewayPorts, HostbasedAuthentication, PasswordAuthentication, PermitOpen, PermitRootLogin, PubkeyAuthentication, RhostsRSAAuthentication, RSAAuthentication, X11DisplayOffset, X11Forwarding, and X11UseLocalHost.
- The maximum number of Group Match criteria arguments is 256.

Guideline: User and group names are typically not case sensitive on z/OS systems. However, when matching user and group names for this keyword and for related keywords (such as the **sshd_config** keywords AllowGroups, AllowUsers, DenyGroups and DenyUsers), the user and group names must be in the same alphabetical case as is stored in the user database, group database and user ID alias table (for example, USERIDALIASTABLE).

Example:

```
AllowTcpForwarding no

Match Address 192.168.32.*,127.0.0.1
        AllowTcpForwarding yes
        GatewayPorts no

Match User bar,baz
        AllowTcpForwarding yes

Match Host t*
        AllowTcpForwarding yes
```

MaxAuthTries
Specifies the maximum number of authentication attempts permitted per connection. When the number of failures reaches half this value, additional failures are logged. The default is 6.

Password authentication failures are always logged.

MaxStartups
Specifies the maximum number of concurrent unauthenticated connections to the SSH daemon. Additional connections will be dropped until authentication succeeds or the LoginGraceTime expires for a connection. The default is 10.

Alternately, random early drop can be enabled by specifying the three colon separated values "start:rate:full" (for example, "10:30:60"). **sshd** will refuse connection attempts with a probability of "rate/100" (30%, in the example) if there are currently "start" (10) unauthenticated connections. The

probability increases linearly and all connection attempts are refused if the number of unauthenticated connections reaches "full" (60).

PAMAuthenticationViaKbdInt
Not supported on z/OS UNIX. Specifies whether PAM challenge-response authentication is allowed. This option allows the use of most PAM challenge-response authentication modules, but it will allow password authentication regardless of whether PasswordAuthentication is enabled.

PasswordAuthentication
Specifies whether password authentication is allowed. The argument must be set to "yes" or "no". The default is "yes". Password authentication checks a user-supplied password or password phrase.

PermitEmptyPasswords
Specifies whether the server allows login to accounts with empty password strings when password authentication is allowed. The default is "no".

Guideline: Set this keyword to "no" for security reasons. However, empty passwords can be allowed by setting up a SURROGAT class. The MVS identity running **sshd** requires READ access to the SURROGAT class profile, BPX.SRV.*uuuuuuuu* (where *uuuuuuuu* is the MVS userid for each user who is permitted to log in with an empty password.) This allows any user to login to user ID *uuuuuuuu* without a password.

PermitOpen
Specifies the destinations to which TCP port forwarding is permitted. The forwarding specification must be one of the following forms:

```
PermitOpen host:port
PermitOpen IPv4_addr:port
PermitOpen [IPv6_addr]:port
```

Multiple forwards can be specified by separating them with white space. An argument of "any" can be used to remove all restrictions and permit any forwarding requests. By default, all port forwarding requests are permitted.

Restriction: The maximum number of forwards that can be specified is 100.

PermitRootLogin
Specifies whether a superuser (root) can login using **ssh**. The argument must be "yes" (default), "without-password", "forced-commands-only", or "no".

If this option is set to "without-password", password authentication is disabled for superusers.

If this option is set to "forced-commands-only", superuser login with public key authentication will be allowed, but only if the Authorized Keys File "command=" option has been specified (which may be useful for taking remote backups even if superuser login is normally not allowed). All other authentication methods are disabled for superusers.

If this option is set to "no", a superuser is not allowed to login.

PermitTunnel
Not supported on z/OS UNIX. Specifies whether tunnel device forwarding is allowed. The argument must be "yes", "point-to-point" (layer 3), "ethernet" (layer 2), or "no". Specifying "yes" permits both "point-to-point" and "ethernet". The default is "no".

PermitUserEnvironment
Specifies whether the ~/.ssh/environment and environment= options in ~/.ssh/authorized_keys are processed by **sshd**. The default is "no". Enabling environment processing might enable users to bypass access restrictions in some configurations using mechanisms such as LD_PRELOAD.

The user's environment variables are processed after authentication and after the **sshd_config** keyword AcceptEnv is processed. As a result, the values of the user's environment variables might overwrite the results of the previous environment variable processing.

PidFile
Specifies the file that contains the process ID of the **sshd** daemon. The default is /var/run/sshd.pid.

Port Specifies the port number that **sshd** listens on. The default is 22. Multiple options of this type are permitted. See also ListenAddress.

PrintLastLog
Not supported on z/OS UNIX. Specifies whether **sshd** should print the date and time of the last user login when a user logs in interactively. The default is "no". This option only returns information if your system supports lastlog data, such as with a wtmp or wtmpx file.

PrintMotd
Specifies whether **sshd** should print /etc/motd when a user logs in interactively. (On some systems, the shell, /etc/profile, or equivalent also prints /etc/motd.) The default is "yes". For more information about the use of /etc/motd during the login process, see “Login process” on page 119.

Protocol
Specifies the protocol versions **sshd** should support. The possible values are "1" and "2". Multiple versions must be comma-separated. The default is "2".

PubkeyAuthentication
Specifies whether public key authentication is allowed. The default is "yes".

Restriction: This option applies to protocol version 2 only.

RhostsAuthentication
Specifies whether authentication using rhosts or /etc/hosts.equiv files is sufficient. Normally, this method should not be permitted, because it is insecure. RhostsRSAAuthentication should be used instead, because it performs RSA-based host authentication in addition to normal rhosts or /etc/hosts/.equiv authentication. The default is "no".

Restrictions:

1. This option apples to protocol version 1 only.
2. RhostsAuthentication cannot be used with privilege separation.

Note: This option was removed from the OpenSSH open source base distribution.

RhostsRSAAuthentication
Specifies whether rhosts or /etc/hosts.equiv authentication together with successful RSA host authentication is allowed. The default is "no".

Restriction: This option applies to protocol version 1 only.

RSAAuthentication
Specifies whether pure RSA authentication is allowed.

Restriction: This option applies to protocol version 1 only.

ServerKeyBits
Determines the number of bits in the ephemeral protocol version 1 server key. The minimum value is 512 and the default is 768.

StrictModes
Specifies whether **sshd** should check file modes and ownership of the user's files and home directory before accepting login. This is normally desirable in case users inadvertently leave their directory or files world-writable. The default is "yes".

Specifically, StrictModes checks that the following files, directories, and component path names are owned by the current user or superuser and that they are not group or world-writable:

- User's home directory
- User's `.rhosts` and `.shosts` files
- User's authorized keys file
- User's known hosts file

Subsystem
Configures an external subsystem (such as file transfer daemon) in protocol version 2. Arguments should be a subsystem name and a command with optional arguments to execute upon subsystem request.

The command **/usr/lib/ssh/sftp-server** implements the **sftp** file transfer subsystem. Alternatively, the name "internal-sftp" implements an in-process **sftp** server. Using the in-process sftp-server might simplify configurations that use the ChrootDirectory keyword to force a different file system root on clients. You can specify **sftp-server** options with the "internal-sftp" command by separating the options with blank spaces.

By default, no subsystems are defined. User-defined (non-builtin) subsystems are only supported between z/OS and z/OS. See “Limitations” on page 157 for more information.

SyslogFacility
Gives the facility code that is used when logging messages from **sshd**. The possible values are: DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. If **sshd** is run in debug mode (invoked with **–d**), logging goes to stderr instead of the syslog. The default is AUTH.

For more information about these log facilities, see the syslog daemon section in *z/OS Communications Server: IP Configuration Reference*.

TCPKeepAlive
Specifies whether the system should send TCP keepalive messages to the other side. If they are sent, a lost network connection or stopping of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily, and some people find it annoying. On the other hand, if keepalives are not sent, sessions may hang indefinitely on the server, leaving ghost users and consuming server resources. The default is "yes" (to send TCP keepalive messages), and the server will notice if the network goes down or the client host crashes. This option avoids infinitely hanging sessions. To disable TCP keepalive messages, set the value to "no".

UseDNS
Specifies whether **sshd** should look up the remote host name and check that the resolved host name for the remote IP address maps back to the same IP address. The default is "yes".

UseLogin
Specifies whether **login** is used for interactive login sessions. **login** is never used for remote command execution. If UseLogin is enabled, X11 forwarding will be disabled because **login** does not know how to handle xauth cookies. If UsePrivilegeSeparation is specified, UsePrivilegeSeparation is disabled after authentication. The default is "no".

UsePAM
Not supported on z/OS UNIX. Enables PAM authentication (via challenge-response) and session set up. The default is "no".

UsePrivilegeSeparation
Specifies whether **sshd** separates privileges by creating an unprivileged child process to deal with incoming network traffic. After successful authentication, another process will be created that has the privilege of the authenticated user. The goal of privilege separation is to prevent privilege escalation by containing any corruption within the unprivileged processes. The default is "yes".

VerifyReverseMapping
This keyword is supported for compatibility with versions of OpenSSH before 3.8.1p1. On systems using OpenSSH 3.8.1p1 or later, use the keyword UseDNS.

Specifies whether **sshd** should try to verify the remote host name and check that the resolved host name for the remote IP address maps back to the same IP address. The default is "yes".

X11DisplayOffset
Specifies the first display number available for **sshd**'s X11 forwarding. This prevents **sshd** from interfering with real X11 servers. The default is "10".

X11Forwarding
Specifies whether X11 forwarding is permitted. Disabling X11 forwarding does not improve general z/OS security, because users can install their own forwarders. X11 forwarding is automatically disabled if UseLogin is enabled. The default is "no".

X11UseLocalhost
Specifies whether **sshd** should bind the X11 forwarding server to the loopback address or to the wildcard address. By default **sshd** binds the forwarding server to the loopback address and sets the hostname part of the DISPLAY environment variable to *localhost*. This prevents remote hosts from connecting to the fake display. However, some X11 clients may not function with this configuration. X11UseLocalhost can be set to "no" to specify that the forwarding server should be bound to the wildcard address. The argument must be "yes" (default) or "no".

XAuthLocation
Specifies the location of the xauth program. The default is `/usr/X11R6/bin/xauth`.

Limitations

User-defined subsystems are only supported between z/OS and z/OS. This is due to a limitation in the SECSH protocol with regards to EBCDIC platforms; for information about the IETF SECSH RFCs and internet drafts, see Appendix C,

"RFCs and Internet drafts," on page 339. User-defined subsystems are specified by using the **sshd_config** subsystem keyword. Only the built-in **sftp** subsystem is supported for transfers between all platforms.

Time formats

sshd command-line arguments and configuration file options that specify time can be expressed using a sequence of the form: *time[qualifier]* where *time* is a positive integer value and *qualifier* is one of the following:

- <none> seconds
- s | S seconds
- m | M minutes
- h | H hours
- d | D days
- w | W weeks

Each member of the sequence is added together to calculate the total time value.

Time format examples:

```
600     600 seconds (10 minutes)
10m     10 minutes
1h30m   1 hour 30 minutes (90 minutes)
```

Files

/etc/ssh/sshd_config
Contains configuration data for **sshd**. This file should be writable by superuser only, but it is recommended (though not necessary) that it be world-readable.

Related information

sshd

Authors

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0. Niels Provos and Markus Friedl contributed support for privilege separation.

zos_sshd_config — z/OS-specific OpenSSH daemon configuration file

Description

z/OS obtains z/OS-specific daemon configuration data in the following order:

1. Command-line specification using the **sshd -o** option.
2. Configuration file specified with the environment variable _ZOS_SSHD_CONFIG. The default is `/etc/ssh/zos_sshd_config`. For each keyword, the first obtained value is used.

Restriction: z/OS-specific keywords cannot be specified in the **sshd_config** configuration files such as the system-wide configuration file (`/etc/ssh/sshd_config`) or the user-defined configuration file specified with the **sshd -f** option.

File format

The **zos_sshd_config** configuration file views empty lines and lines starting with # as comments. Configuration options can be specified using two different formats.

- The first format is the keyword argument pair separated by white space.
- The second format is the keyword argument pair separated with exactly one "=" and optional white space. This format avoids the need to quote white space when specifying configuration options using the **sshd -o** option. Arguments can optionally be enclosed in double quotes (") in order to represent arguments containing spaces.

  Example:

  ```
  keyword argument
  keyword=argument
  ```

Keywords are not case sensitive and arguments are case sensitive. Following are the possible keywords:

HostKeyRingLabel
: Specifies the key ring owner, name of the key ring and certificate label within the key ring containing a private host key used by OpenSSH. The key ring can be real or virtual, and certificate labels can contain embedded blanks. The key ring and the certificate connected to the key ring were created in the server authentication setup, which are described in “Steps for setting up user authentication when keys are stored in key rings” on page 68. One or more blanks separate the key ring name from the certificate label. The host private key is read from this key ring before HostKey files are checked. The default is to use only the HostKey file (or files).

  It is possible to have multiple host key files and key ring certificates in configuration files. If both host key files and key ring certificates are used, the key ring certificates are tried first. Only the first key found of each type (for example, RSA, DSA, or RSA1) is used. The maximum combined number of host key files and key ring certificates that can be specified is 256.

  The option value must be surrounded by double quotes.

  Example: An example of this option in the **zos_sshd_config** file for a key ring named 'SSHDring' that is owned by SSHDAEM and a certificate labeled 'my label with blanks' is as follows:

  ```
  HostKeyRingLabel="SSHDAEM/SSHDring my label with blanks"
  ```

  If the option is specified as a command-line option, you might need to escape the double quote characters that surround the argument value:

  ```
  -o HostKeyRingLabel="\"SSHDAEM/SSHDring my label with blanks\""
  ```

Match Introduces a conditional block. If all of the criteria on the Match line are satisfied, the keywords on the following lines override those set in the global section of the config file, until either another Match line or the end of the file.

  Rule: Global settings must be placed before the first Match block.

  The arguments to Match are one or more criteria-pattern pairs. The available criteria are User, Group, Host, and Address. The match patterns can consist of single entries or comma-separated lists and can use the wildcard and negation operators described in the **ssh_config** section “Patterns” on page 140.

Restrictions:
1. Only the ServerSMF keyword can be used on the line following a Match keyword.
2. The maximum number of Group Match criteria arguments is 256.

Guideline: User and group names are typically not case sensitive on z/OS systems. However, when matching user and group names for this keyword, the user and group names must be in the same alphabetical case as is stored in the user database, group database and user ID alias table (for example, USERIDALIASTABLE).

For example:

```
ServerSMF none

Match Address 192.168.32.*,127.0.0.1
    ServerSMF TYPE119_U83

Match User bar,baz
  ServerSMF TYPE119_U84

Match Host t*
  ServerSMF TYPE119_U83
```

ServerSMF
Specifies whether to collect server SMF records. The argument must be set to "TYPE119 _U83", "TYPE119_U84" or "none". The default is "none". If set to "TYPE119 _U83" or "TYPE119_U84", SMF Type 119 login failure records (subtype 98) are collected as well as server transfer completion records (subtype 96) for the **sftp** and **scp** commands. SMF record exit IEFU83 receives control for "TYPE119_U83". SMF record exit IEFU84 receives control for "TYPE119_U84".

Environment variable

_ZOS_SSHD_CONFIG
Specifies the path name of the user-defined **zos_sshd_config** configuration file. The default is `/etc/ssh/zos_sshd_config`. See "File format" on page 159 for the available keywords. The recommended permissions of the specified file are read/write for the user and not accessible by others.

Files

/etc/ssh/zos_sshd_config
z/OS-specific system-wide daemon configuration file. This file must be world-readable but writable only by a superuser.

Related information

scp, **sftp**, **sftp-server**, **sshd**

Other OpenSSH files

moduli — System moduli file

Description

The `/etc/ssh/moduli` file contains the system-wide Diffie-Hellman prime moduli for **sshd**. Each line in this file contains the following fields: Time, Type, Tests, Tries, Size, Generator, Modulus. The fields are separated by white space (tab or blank). The file is searched for moduli that meet the appropriate Time, Size and Generator

criteria. When more than one meet the criteria, the selection should be weighted toward newer moduli, without completely disqualifying older moduli.

File format

Time: yyyymmddhhmmss
Specifies the system time that the line was appended to the file. The value 00000000000000 means unknown (historic).

Type: decimal
Specifies the internal structure of the prime modulus.

0 Unknown; often learned from peer during protocol operation, and saved for later analysis.

1 Unstructured; a common large number.

2 Safe (p = 2q + 1); meets basic structural requirements.

3 Schnorr.

4 Sophie-Germaine (q = (p-1)/2); usually generated in the process of testing safe or strong primes.

5 Strong; useful for RSA public key generation.

Tests: decimal (bit field)
Specifies the methods used in checking for primality. Usually, more than one test is used.

0 Not tested; often learned from peer during protocol operation, and saved for later analysis.

1 Composite; failed one or more tests. In this case, the highest bit specifies the test that failed.

2 Sieve; checked for division by a range of smaller primes.

4 Miller-Rabin.

8 Jacobi.

16 Elliptic Curve.

Tries: decimal
Depends on the value of the highest valid Test bit, where the method specified is:

0 Not tested (always zero).

1 Composite (irrelevant).

2 Sieve; number of primes sieved. Commonly on the order of 32,000,000.

4 Miller-Rabin; number of M-R iterations. Commonly on the order of 32 to 64.

8 Jacobi; unknown (always zero).

16 Elliptic Curve; unused (always zero).

Size: decimal
Specifies the number of significant bits.

Generator: hex string
Specifies the best generator for a Diffie-Hellman exchange. 0 = unknown or variable such as 2, 3, or 5.

Modulus: hex string
The prime modulus.

Related information

sshd

Chapter 11. OpenSSH files Quick Reference

This topic provides a quick reference to OpenSSH files. It includes the following sections:

- "Configuration files"
- "Program-generated files"
- "Administrator-generated user files" on page 164
- "User-generated files" on page 164

Configuration files

Table 16 lists the configuration files that must be copied into the `/etc/ssh` directory. It also lists the owner and permissions that are needed for each file.

Table 16. Configuration files to copy into /etc (including permissions)

File	Copied to	Description	Permissions	Owner
/samples/moduli	/etc/ssh/moduli	Contains Diffie-Hellman groups for **sshd**. See "moduli" on page 160.	644	UID(0)
/samples/ssh_prng_cmds	/etc/ssh/ssh_prng_cmds	Contains commands for gathering entropy	644	UID(0)
/samples/ssh_config	/etc/ssh/ssh_config	OpenSSH client configuration file. See "ssh_config" on page 129.	644	UID(0)
/samples/sshd_config	/etc/ssh/sshd_config	OpenSSH daemon configuration file. See "sshd_config" on page 144.	600	UID(0)
/samples/zos_ssh_config	/etc/ssh/zos_ssh_config	z/OS-specific system-wide OpenSSH client configuration file. See "zos_ssh_config" on page 141.	644	UID(0)
/samples/zos_sshd_config	/etc/ssh/zos_sshd_config	z/OS-specific OpenSSH daemon configuration file. See "zos_sshd_config" on page 158.	600	UID(0)

Program-generated files

Table 17 lists the files generated by OpenSSH. It also lists the owner and permissions that are set when the files are generated.

Table 17. Program-generated files (including permissions)

File	Produced by	Description	Permissions	Owner
~/.ssh/prng_seed	**ssh-rand-helper**	Seed file used by **ssh-rand-helper**	600	User
/var/run/sshd.pid	**sshd**	**sshd** daemon process ID	644	UID(0)
/var/run/sshd.mm.XXXXXXXX	**sshd**	Temporary files used for compression with privilege separation	600	UID(0)

Administrator-generated user files

Table 18 lists the files generated by the administrator. It also lists the owner and permissions that are set when the files are generated.

Table 18. Administrator-generated files (including permissions)

File	Produced by	Description	Permissions	Owner
/etc/ssh/sshrc	Administrator	Optional host-specific initialization script	644	UID(0)
/etc/ssh/ssh_host_key	**ssh-keygen**	Host private key file	600	UID(0)
/etc/ssh/ssh_host_dsa_key	**ssh-keygen**	Host private DSA key file	600	UID(0)
/etc/ssh/ssh_host_rsa_key	**ssh-keygen**	Host private RSA key file	600	UID(0)
/etc/ssh/ssh_host_key.pub	**ssh-keygen**	Host public key file	644	UID(0)
/etc/ssh/ssh_host_dsa_key.pub	**ssh-keygen**	Host public DSA key file	644	UID(0)
/etc/ssh/ssh_host_rsa_key.pub	**ssh-keygen**	Host public RSA key file	644	UID(0)
/etc/ssh/ssh_known_hosts	Administrator (possibly by using **ssh-keyscan**)	Public keys for remote hosts allowed by system	644	UID(0)
/etc/hosts.equiv	Administrator	Not recommended. Hosts listed in `.rhosts` authentication.	644	UID(0)
/etc/ssh/shosts.equiv	Administrator	Not recommended. Hosts list used in **ssh** host-based authentication.	644	UID(0)
/etc/nologin	Administrator	If it exists, prevents non-superuser **sshd** login and outputs contents to user.	644	UID(0)

User-generated files

Table 19 lists the files that can be generated by the user. It also lists the owner and permissions that are set when the files are created.

Table 19. User-generated files (including permissions)

File	Produced by	Description	Permissions	Owner
~/.ssh/known_hosts	Remote host key added to the file when user connects to an unknown host.	Public keys for remote hosts that users can communicate with.	600	User
~/.ssh/authorized_keys	User	Public keys that can be used to log in to user's account. Copied from ~/.ssh/*.pub files of this user's accounts on other (remote) systems.	600	User
~/.rhosts	User	Not recommended. Hosts and users lists to which user can login without password.	600	User

Table 19. User-generated files (including permissions) (continued)

File	Produced by	Description	Permissions	Owner
~/.shosts	User	Not recommended. Hosts and users lists that users can login (via **sshd** only) without password.	600	User
~/.ssh/config	User	Per-user OpenSSH client configuration file	600	User
~/.ssh/zos_user_ssh_config	User	z/OS-specific per-user OpenSSH client configuration file	600	User
~/.ssh/environment	User	User's environment variable initialization at **ssh** login	600	User
~/.ssh/rc	User	User's initialization script at **ssh** login	600	User
~/.ssh/identity	**ssh-keygen**	User private key file (protocol version 1)	600	User
~/.ssh/id_dsa	**ssh-keygen**	User private DSA key file	600	User
~/.ssh/id_rsa	**ssh-keygen**	User private RSA key file	600	User
~/.ssh/identity.pub	**ssh-keygen**	User public key (protocol version 1)	644	User
~/.ssh/id_dsa.pub	**ssh-keygen**	User public DSA key	644	User
~/.ssh/id_rsa.pub	**ssh-keygen**	User public RSA key	644	User

Chapter 12. SMF Type 119 records for OpenSSH

This topic describes the SMF Type 119 records collected for OpenSSH servers and clients.

Common SMF Type 119 record format

C-level macros for mapping OpenSSH SMF Type 119 records can be found in `/samples/ssh_smf.h`. Assembler mappings can be found in FOTSMF77 in SYS1.MACLIB.

All Type 119 SMF records are in the format shown in Table 20. For a list of record subtypes that OpenSSH supports, see "SMF 119 record subtypes for OpenSSH" on page 168.

Table 20. Records types and subtype information

Offset	Name	Length	Format	Description
0(x'0')	Standard header	24	Binary	SMF system header
0(x'0')	SMF_119SSH_HDLength	2	Binary	SMF record length
2(x'2')	SMF_119SSH_HDSegDesc	2	Binary	Segment descriptor
4(x'4')	SMF_119SSH_HDFlags	1	Binary	Record flags
5(x'5')	SMF_119SSH_HDType	1	Binary	Record type; is set to 119 (x'77')
6(x'6')	SMF_119SSH_HDTime	4	Binary	SMF system time stamp (is local time)
10(x'A')	SMF_119SSH_HDDate	4	Packed	SMF system date (is local date)
14(x'D')	SMF_119SSH_HDSID	4	EBCDIC	SMF system ID
18(x'12')	SMF_119SSH_HDSSI	4	EBCDIC	SMF subsystem ID
22(x'16')	SMF_119SSH_HDSubType	2	Binary	Record subtype
24(x'18')	Self-defining section		Binary	This section indicates how many sections follow and their location in the record.
...	TCP/IP identification section for OpenSSH	64	Binary	This section is present in every record; it describes the TCP/IP stack that issued the record. Its location and size are indicated by the self-defining section.
...	Record-specific data section 1	...	Binary	First record-specific data section. Its location and size are indicated by the self-defining section.
...	Record-specific data section 1, second entry	...	Binary	The self-defining section indicates how many occurrences of each record-specific data section are present in the record.
...	Record-specific data section 2 (optional)	...	Binary	Second record-specific data section.
...	...	...	Binary	...
...	Record-specific data section *n*, first entry (optional)	...	Binary	Last record-specific data section. The self-defining section indicates how many types of data sections there are.

z/OS MVS System Management Facilities (SMF) contains information about SMF headers. For more information about the other sections, see the section on SMF Type 119 records in *z/OS Communications Server: IP Configuration Reference*.

SMF 119 record subtypes for OpenSSH

OpenSSH collects SMF Type 119 records for file transfer activity and login failure information. You can control the collection of these records by using the configuration keywords ClientSMF and ServerSMF in z/OS-specific client and daemon configuration files, respectively. These keywords also indicate whether system-wide SMF record exit IEFU83 or IEFU84 receives control. For more information about those keywords, see “zos_ssh_config” on page 141 and “zos_sshd_config” on page 158.

The specified SMF record exit receives control before each record is written to the SMF data set. A return code from this exit indicates whether the system is to suppress the current SMF record. The parameter passed to this exit is the SMF record to be written. See *z/OS MVS System Management Facilities (SMF)* for more information.

All the records described in this topic are written using record type x'77' (format 119), and record subtype values, at offset 22(x'16') in the SMF record header, are used to uniquely identify the type of record being collected as well as describing the values that will be seen in the SMF_119SSH_TI_Comp and SMF_119SSH_TI_Reason fields of the TCP/IP identification section. Table 21 correlates the subtypes collected by OpenSSH to the type of record being produced.

Table 21. OpenSSH SMF Type 119 record subtype information and record type

Record subtype	Description	Component	Reason
96(x'60')	Server transfer completion record	SFTPS or SCPS	Event
97(x'61')	Client transfer completion record	SFTPC or SCPC	Event
98(x'62')	Login failure record	SSHD	Event

Additional SMF Type 119 subtype records are provided by z/OS Communications Server and are described in *z/OS Communications Server: IP Configuration Reference*.

Standard data format concepts

The following concepts apply to standard data formats:

- Unless specified otherwise, all times are indicated in units of 1/100 seconds since midnight UTC/GMT (Universal Time, Coordinated/Greenwich Mean Time).
- All dates are indicated in packed binary-coded decimal (BCD) format, with digits x'01yydddF'. If no data is available, a date of x'0000000F' is written
- Interval durations are specified in units of 1/100 seconds.
- All IP addresses are in 128-bit IPv6 format. IPv4 addresses are reported in IPv4-mapped form where the 4-byte IPv4 address is preceded by 12 bytes, the first 10 of which are 0, and the last two of which are 'FF'x. IPv6 addresses appears in numeric form.
- Unless specified otherwise, all path names are absolute path names.

Common TCP/IP identification section for OpenSSH

Table 22 shows a section that is present in every SMF Type 119 record. It identifies the system and stack information associated with the SMF record.

Table 22. Common TCP/IP identification section for OpenSSH

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_TI_SYSName	8	EBCDIC	System name from SYSNAME in IEASYSxx
8(x'8')	SMF_119SSH_TI_SysplexName	8	EBCDIC	Sysplex name from SYSPLEX in COUPLExx
16(x'10')	SMF_119SSH_TI_Stack	8	EBCDIC	TCP/IP stack name
24(x'18')	SMF_119SSH_TI_ReleaseID	8	EBCDIC	z/OS release identifier
32(x'20')	SMF_119SSH_TI_Comp	8	EBCDIC	OpenSSH subcomponent (right-padded with blanks): **SFTPS** **sftp** server **SFTPC** **sftp** client **SCPS** **scp** server **SCPC** **scp** client **SSHD** **sshd** daemon
40(x'28')	SMF_119SSH_TI_ASName	8	EBCDIC	Started task qualifier or address space name of address space that writes this SMF record
48(x'30')	SMF_119SSH_TI_UserID	8	EBCDIC	User ID of security context under which this SMF record is written
56(x'38')	Reserved	2	Binary	Reserved
58(x'3A')	SMF_119SSH_TI_ASID	2	Binary	ASID of address space that writes this SMF record
60(x'3C')	SMF_119SSH_TI_Reason	1	Binary	Reason for writing this SMF record **x'08'** Event record
61(x'3D')	SMF_119SSH_TI_RecordID	1	Binary	Record ID
61(x'3E')	Reserved	2	EBCDIC	Reserved

Common security section for OpenSSH

Table 23 shows a section that is present in every SMF Type 119 record. It identifies the security information associated with the SMF record.

Table 23. Common security section

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_SSHV	16	EBCDIC	OpenSSH version
16(x'10')	SMF_119SSH_SSLV	32	EBCDIC	OpenSSL version
48(x'30')	SMF_119SSH_ZlibV	16	EBCDIC	zlib version
64(x'40')	SMF_119SSH_ProtoV	8	EBCDIC	Protocol version (right-padded with blanks): **'SSHV1'** Protocol version 1 **'SSHV2'** Protocol version 2
72(x'48')	SMF_119SSH_AuthMethod	2	Binary	Authentication method being used: **x'0000'** Unknown **x'0001'** None **x'0002'** Password **x'0003'** Public key **x'0004'** Host-based **x'0005'** Rhosts **x'0006'** RhostsRSA **x'0007'** RSA **x'0008'** Keyboard-interactive **x'0009'** Challenge-response **x'000A'** Control socket [1]

Table 23. Common security section (continued)

Offset	Name	Length	Format	Description
74(x'4A')	SMF_119SSH_Cipher	2	Binary	Cipher type being used: **x'0000'** Unknown **x'0001'** None Possible values when protocol version 1: **x'0002'** 3DES **x'0003'** Blowfish **x'0004'** DES Possible values when protocol version 2: **x'0005'** 3des-cbc **x'0006'** blowfish-cbc **x'0007'** cast128-cbc **x'0008'** arcfour128 **x'0009'** arcfour256 **x'000A'** arcfour **x'000B'** aes128-cbc **x'000C'** aes192-cbc **x'000D'** aes256-cbc **x'000E'** aes128-ctr **x'000F'** aes192-ctr **x'0010'** aes256-ctr **x'0011'** rijndael-cbc@lysator.liu.se **x'0012'** acss@openssh.org
76(x'4C')	SMF_119SSH_MAC	2	Binary	MAC algorithm being used: **x'0000'** Unknown **x'0001'** None (protocol version 1) **x'0002'** hmac-md5 **x'0003'** hmac-sha1 **x'0004'** umac-64@openssh.com **x'0005'** hmac-ripemd160 **x'0006'** hmac-sha1-96 **x'0007'** hmac-md5-96 **x'0008'** hmac-ripemd160@openssh.com
78(x'4E')	SMF_119SSH_COMP	2	Binary	Compression method being used: **x'0000'** Unknown **x'0001'** None (no) **x'0002'** zlib (yes) **x'0003'** zlib@openssh.com (delayed)

Notes:

1. When the authentication method being used is Control Socket and the **ssh** connection information cannot be collected from the control socket, the EBCDIC fields are set to blanks and the binary fields are set to x'0000' Unknown.

Server transfer completion record (subtype 96)

The server transfer completion records are collected when the **sftp-server** (regular or "internal-sftp") or the server side of **scp** completes processing of one of the following file transfer subcommands:

- Creating, uploading, downloading, renaming or removing files
- Creating and removing directories
- Changing the file permissions, UIDs, or GIDs
- Creating symbolic links

For **scp**, only file downloading or uploading apply. A common format for the record is used for each **sftp** file transfer operation, so the record contains an indication of which subcommand was performed.

See Table 22 on page 169 for the contents of the TCP/IP identification section. For the server transfer completion record, the TCP/IP identification section indicates either SFTPS (**sftp-server**) or SCPS (server side of **scp**) as the OpenSSH subcomponent and x'08' (event record) as the record reason.

See Table 23 on page 169 for the contents of the security section.

Table 24 shows the server transfer completion record self-defining section.

Table 24. Server transfer completion record self-defining section

Offset	Name	Length	Format	Description
0(x'0')	Standard SMF Header	24	Reserved	Standard SMF header, where the record subtype is 96 (x'60')
Self-defining section				
24(x'18')	SMF_119SSH_SDTRN	2	Binary	Number of triplets in this record (6)
26(x'1A')	Reserved	2	Binary	Reserved
28(x'1C')	SMF_119SSH_IDOff	4	Binary	Offset to TCP/IP identification section
32(x'20')	SMF_119SSH_IDLen	2	Binary	Length of TCP/IP identification section
34(x'22')	SMF_119SSH_IDNum	2	Binary	Number of TCP/IP identification sections
36(x'24')	SMF_119SSH_S1Off	4	Binary	Offset to security section
40(x'28')	SMF_119SSH_S1Len	2	Binary	Length of security section
42(x'2A')	SMF_119SSH_S1Num	2	Binary	Number of security sections
44(x'2C')	SMF_119SSH_S2Off	4	Binary	Offset to server transfer completion section
48(x'30')	SMF_119SSH_S2Len	2	Binary	Length of server transfer completion section
50(x'32')	SMF_119SSH_S2Num	2	Binary	Number of server transfer completion sections
52(x'34')	SMF_119SSH_S3Off	4	Binary	Offset to server host name section
56(x'38')	SMF_119SSH_S3Len	2	Binary	Length of server host name section
58(x'3A')	SMF_119SSH_S3Num	2	Binary	Number of server host name sections
60(x'3C')	SMF_119SSH_S4Off	4	Binary	Offset to server first associated path name section
64(x'40')	SMF_119SSH_S4Len	2	Binary	Length of server first associated path name section
66(x'42')	SMF_119SSH_S4Num	2	Binary	Number of server first associated path name sections
68(x'44')	SMF_119SSH_S5Off	4	Binary	Offset to server second associated path name section
72(x'48')	SMF_119SSH_S5Len	2	Binary	Length of server second associated path name section
74(x'4A')	SMF_119SSH_S5Num	2	Binary	Number of server second associated path name sections

Table 25 on page 172 shows the server transfer completion specific section of this SMF record.

Table 25. Server transfer completion record specific section

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FSOper	1	Binary	**sftp** subcommand code (for **scp**, only get and put apply): **x'01'** rmdir **x'02'** rm **x'03'** rename **x'04'** get **x'05'** put **x'06'** chmod **x'07'** chown or chgrp **x'08'** mkdir **x'09'** symlink
1(x'1')	Reserved	3	EBCDIC	Reserved
4(x'4')	SMF_119SSH_FSCmd	4	EBCDIC	**sftp** subcommand (the values are right-padded with blanks, and for **scp**, only GET and PUT apply): **RMD** Remove directory **RM** Remove file **RENM** Rename file **GET** Download file from the server **PUT** Upload file to the server **CHMD** Change file permission bits **CHOW** Change file owner or group **MKD** Create directory **SLNK** Create symbolic link
8(x'8')	SMF_119SSH_FSRIP	16	Binary	Remote IP address (client)
24(x'18')	SMF_119SSH_FSLIP	16	Binary	Local IP address (server)
40(x'28')	SMF_119SSH_FSRPort	2	Binary	Remote port number (client)
42(x'2A')	SMF_119SSH_FSLPort	2	Binary	Local port number (server)
44(x'2C')	SMF_119SSH_FSSUser	8	EBCDIC	Client User ID on server
52(x'34')	SMF_119SSH_FSTType	1	EBCDIC	Data transfer type: **A** ASCII **B** Binary
53(x'35')	SMF_119SSH_FSMode	1	EBCDIC	Transfer mode: **C** Compressed **S** Stream
54(x'36')	Reserved	2	Binary	Reserved
56(x'38')	SMF_119SSH_FSSTime	4	Binary	Transmission start time of day
60(x'3C')	SMF_119SSH_FSSDate	4	Packed	Transmission start date
64(x'40')	SMF_119SSH_FSETime	4	Binary	Transmission end time of day
68(x'44')	SMF_119SSH_FSEDate	4	Packed	Transmission end date
72(x'48')	SMF_119SSH_FSDur	4	Binary	File transmission duration in units of 1/100 seconds
76(x'4C')	SMF_119SSH_FSBytes	8	Binary	Transmission byte count; 64-bit integer
84(x'54')	SMF_119SSH_FSStat	4	EBCDIC	Server execution status (right-padded with blanks): **OK** Success **FAIL** Failure
88(x'58')	SMF_119SSH_FSCH1	8	Binary	Previous read/write/execute permissions of owner/group/other (in octal format) when **chmod** is used or the previous UID when **chown** or **chgrp** is used.
96(x'60')	SMF_119SSH_FSGP1	8	Binary	Previous GID when **chown** or **chgrp** is used.
104(x'68')	SMF_119SSH_FSCH2	8	Binary	New read/write/execute permissions of owner/group/other (in octal) when **chmod** is used or the new UID when **chown** or **chgrp** is used.
112(x'70')	SMF_119SSH_FSGP2	8	Binary	New GID when **chown** or **chgrp** is used.

Table 26 shows the host name section for the server transfer completion record.

Table 26. Server transfer completion record section: Host name

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FSHostname	*n*	EBCDIC	Host name

Table 27 shows the first associated path name section for the server transfer completion record. This section represents the server z/OS UNIX path name associated with the **sftp** or **scp** operation.

Table 27. Server transfer completion record section: First associated path name

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FSPath1	*n*	EBCDIC	z/OS UNIX path name associated with the **sftp** or **scp** command. When the subcommand is rename or symlink, this refers to the previous path name.

Table 28 shows the second associated path name section for the server transfer completion record. This section represents the server z/OS UNIX file name associated with the rename or symlink subcommand.

Table 28. Server transfer completion record section: Second associated path name

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FSPath2	*n*	EBCDIC	Second z/OS UNIX path name associated with rename or symlink subcommand. This is the new path name.

Client transfer completion record (subtype 97)

The client transfer completion records are collected when the client side of **sftp** or **scp** completes processing of one of the following file transfer operations:

- Uploading files
- Downloading files

A common format for the record is used for each file transfer operation, so the record contains an indication of which subcommand was performed.

See Table 22 on page 169 for the contents of the TCP/IP identification section. For the client transfer completion record, the TCP/IP identification section indicates either SFTPC (**sftp** client) or SCPC (**scp** client) as the subcomponent and x'08' (event record) as the record reason.

See Table 23 on page 169 for the contents of the security section.

Table 29 shows the client transfer completion record self-defining section.

Table 29. Client transfer completion record self-defining section

Offset	Name	Length	Format	Description
0(x'0')	Standard SMF Header	24	Reserved	Standard SMF header, where the record subtype is 97 (x'61')
Self-defining section				
24(x'18')	SMF_119SSH_SDTRN	2	Binary	Number of triplets in this record (6)
26(x'1A')	Reserved	2	Binary	Reserved

Table 29. Client transfer completion record self-defining section (continued)

Offset	Name	Length	Format	Description
28(x'1C')	SMF_119SSH_IDOff	4	Binary	Offset to TCP/IP identification section
32(x'20')	SMF_119SSH_IDLen	2	Binary	Length of TCP/IP identification section
34(x'22')	SMF_119SSH_IDNum	2	Binary	Number of TCP/IP identification sections
36(x'24')	SMF_119SSH_S1Off	4	Binary	Offset to security section
40(x'28')	SMF_119SSH_S1Len	2	Binary	Length of security section
42(x'2A')	SMF_119SSH_S1Num	2	Binary	Number of security sections
44(x'2C')	SMF_119SSH_S2Off	4	Binary	Offset to client transfer completion section
48(x'30')	SMF_119SSH_S2Len	2	Binary	Length of client transfer completion section
50(x'32')	SMF_119SSH_S2Num	2	Binary	Number of client transfer completion sections
52(x'34')	SMF_119SSH_S3Off	4	Binary	Offset to client transfer completion host name section
56(x'38')	SMF_119SH_S3Len	2	Binary	Length of client transfer completion host name section
58(x'3A')	SMF_119SSH_S3Num	2	Binary	Number of client transfer completion host name section
60(x'3C')	SMF_119SSH_S4Off	4	Binary	Offset to client transfer completion user name section
64(x'40')	SMF_119SSH_S4Len	2	Binary	Length of client transfer completion user name section
66(x'42')	SMF_119SSH_S4Num	2	Binary	Number of client transfer completion user name sections
68(x'44')	SMF_119SSH_S5Off	4	Binary	Offset to client transfer completion associated path name section
72(x'48')	SMF_119SSH_S5Len	2	Binary	Length of client transfer completion associated path name section
74(x'4A')	SMF_119SSH_S5Num	2	Binary	Number of client transfer completion associated path name sections

Table 30 shows the client transfer completion specific record of this SMF record.

Table 30. Client transfer completion record specific section

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FCCmd	4	EBCDIC	**sftp** or **scp** subcommand (right-padded with blanks): **GET** Download file from the server **PUT** Upload file to the server
4(x'4')	SMF_119SSH_FCRIP	16	Binary	Remote IP address (server) [1]
20(x'14')	SMF_119SSH_FCLIP	16	Binary	Local IP address (client) [1]
36(x'24')	SMF_119SSH_FCRPort	2	Binary	Remote port number (server) [1]
38(x'26')	SMF_119SSH_FCLPort	2	Binary	Local port number (client) [1]
40(x'28')	SMF_119SSH_FCLUser	8	EBCDIC	Local user ID
48(x'30')	SMF_119SSH_FCTType	1	EBCDIC	Data transfer type: **A** ASCII **B** Binary
49(x'31')	SMF_119SSH_FCMode	1	EBCDIC	Transfer mode: [2] **C** Compressed **S** Stream
50(x'32')	Reserved	2	Binary	Reserved
52(x'34')	SMF_119SSH_FCSTime	4	Binary	Transmission start time of day
50(x'32')	SMF_119SSH_FCSDate	4	Packed	Transmission start date

Table 30. Client transfer completion record specific section (continued)

Offset	Name	Length	Format	Description
60(x'3C')	SMF_119SSH_FCETime	4	Binary	Transmission end time of day
64(x'40')	SMF_119SSH_FCEDate	4	Packed	Transmission end date
68(x'44')	SMF_119SSH_FCDur	4	Binary	File transmission duration in units of 1/100 seconds
72(x'48')	SMF_119SSH_FCBytes	8	Binary	Transmission byte count; 64-bit integer
80(x'50')	SMF_119SSH_FCStat	4	EBCDIC	Subcommand execution status (right-padded with blanks): **OK** Success **FAIL** Failure

Notes:

1. This field will be set to zero (0) when the Authentication method being used is Control Socket and the **ssh** connection information could not be collected from the control socket.
2. This field will be set to blank when the Authentication method being used is Control Socket and the **ssh** connection information could not be collected from the control socket.

Table 31 shows the client transfer completion host name section.

Table 31. Client transfer completion host name section

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FCHostname	*n*	EBCDIC	Client host name

Table 32 shows the client transfer completion user name section.

Table 32. Client transfer completion user name section

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FCUserID	*n*	EBCDIC	User name used to log into the server[1]

Notes:

1. This field will not be set when the Authentication method being used is Control Socket and the **ssh** connection information could not be collected from the control socket.

Table 33 shows the client transfer completion associated path name section. This section represents the client z/OS UNIX path name associated with the **sftp** or **scp** subcommand.

Table 33. Client transfer completion associated path name section

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_FCPath	*n*	EBCDIC	Client z/OS UNIX path name

Login failure record (subtype 98)

Login failure records are collected after each unsuccessful attempt to log into the **sshd** daemon. A login failure record is collected for each authentication method and attempt that fails. A login failure reason code within the SMF record provides information about the cause of the login failure. Only failures during user authentication are collected with the following exception: records are not collected for a "none" authentication failure if it is the first authentication method attempted.

See Table 22 on page 169 for the contents of the TCP/IP identification section. For the login failure record, the TCP/IP identification section indicates SSHD (**ssh** daemon) as the subcomponent and x'08' (event record) as the record reason.

See Table 23 on page 169 for the contents of the security section.

Table 34 shows the login failure record self-defining section.

Table 34. Login failure record self-defining section

Offset	Name	Length	Format	Description
0(x'0')	Standard SMF Header	24	Reserved	Standard SMF header, where the record subtype is 98 (x'62')
Self-defining section				
24(x'18')	SMF_119SSH_SDTRN	2	Binary	Number of triplets in this record (3)
26(x'1A')	Reserved	2	Binary	Reserved
28(x'1C')	SMF_119SSH_IDOff	4	Binary	Offset to TCP/IP identification section
32(x'20')	SMF_119SSH_IDLen	2	Binary	Length of TCP/IP identification section
34(x'22')	SMF_119SSH_IDNum	2	Binary	Number of TCP/IP identification sections
36(x'24')	SMF_119SSH_S1Off	4	Binary	Offset to security section
40(x'28')	SMF_119SSH_S1Len	2	Binary	Length of security section
42(x'2A')	SMF_119SSH_S1Num	2	Binary	Number of security sections
44(x'2C')	SMF_119SSH_S2Off	4	Binary	Offset to login failure section
48(x'30')	SMF_119SSH_S2Len	2	Binary	Length of login failure section
50(x'32')	SMF_119SSH_S2Num	2	Binary	Number of login failure sections

Table 35 shows the login failure specific section of this SMF record.

Table 35. Login failure specific section

Offset	Name	Length	Format	Description
0(x'0')	SMF_119SSH_LFRIP	16	Binary	Remote IP address
16(x'10')	SMF_119SSH_LFLIP	16	Binary	Local IP address
32(x'20')	SMF_119SSH_LFRPort	2	Binary	Remote port number (client)
34(x'22')	SMF_119SSH_LFLPort	2	Binary	Local port number (server)
36(x'24')	SMF_119SSH_LFUserID	8	EBCDIC	User name (login name) on server
44(x'2C')	SMF_119SSH_LFReason	2	Binary	Login failure reason: **x'0000'** Unexpected authentication failure. **x'0001'** Unexpected authentication change **x'0002'** Password or password phrase is not valid. **x'0003'** User ID has been revoked **x'0004'** User does not have server access **x'0005'** User's file has bad file modes or ownership **x'0006'** Too many failed login attempts **x'0007'** Password error **x'0008'** User ID is unknown. **x'0009'** Root user authentication is not allowed **x'000A'** Empty passwords are not permitted **x'000B'** Authentication method did not exist or was not valid **x'000C'** Key did not exist or was not valid **x'000D'** Host did not exist or was not valid
46(x'2E')	Reserved	2	Binary	Reserved

Chapter 13. Troubleshooting

This topic discusses performance considerations when troubleshooting setup problems. A FAQ (frequently asked questions) section is included as well as information about setting up the syslogd daemon to debug **sshd** problems.

Performance considerations

Various setup problems might affect OpenSSH performance.

XPLINK is not set up

If performance is not ideal, verify that you have set up XPLINK as described in "Setting up the XPLINK environment for use by IBM Ported Tools for z/OS: OpenSSH" on page 14.

DNS is not configured properly

The **ssh** client performs some DNS lookups. If the DNS server is down, some operations might take a while to time out. Verify that the DNS is configured properly. Also verify that the servers in the DNS resolution files (for example, `/etc/resolv.conf`) are working. If the **ssh** command, when run in verbose mode (**-vvv**), seems to be waiting on this line:

```
debug2: ssh_connect: needpriv 0
```

then it is likely that the DNS is not configured properly.

The system might need tuning for z/OS UNIX or OpenSSH

The OpenSSH commands invoke `/usr/lib/ssh/ssh-rand-helper` to gather random data. If your OpenSSH command, when run in verbose mode (**-vvv**), seems to be waiting on this line:

```
debug3: Seeding PRNG from /usr/lib/sssh/ssh-rand-helper
```

then the commands listed in `/etc/ssh/ssh_prng_cmds` and run by **ssh-rand-helper** could be timing out. Run **ssh-rand-helper** manually (from your shell prompt) to see how many and which commands are timing out.

Example:

```
/usr/lib/ssh/ssh-rand-helper -vvv
```

If every command is timing out, look for more tuning tips in *z/OS UNIX System Services Planning* and *z/OS MVS Initialization and Tuning Reference*. Also consider editing your **/etc/ssh/ssh_prng_cmds** file to contain different commands or modifying the _ZOS_SSH_PRNG_CMDS_TIMEOUT environment variable. For more information, see "ssh-rand-helper" on page 115.

Frequently asked questions

1. **The following RACF warning appeared many times on the console while starting ssh. Does that mean that something is wrong?**

```
ICH408I USER(WELLIE1   ) GROUP(SYS1   ) NAME(WELLIE1   )
   CSFRNG CL(CSFSERV )
   INSUFFICIENT ACCESS AUTHORITY
   FROM CSFRNG (G)
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

   If ICSF is installed, random numbers can be generated from hardware (`/dev/random` or `/dev/urandom`) instead of the software algorithm **ssh-rand-helper**. In order to use the ICSF random number generate service, the user ID needs to have read access to the CSFRNG profile. The RACF warning is issued due to lack of access authority. For information about how to authorize the user ID to the CSFRNG profile, see "Using hardware support to generate random numbers" on page 49. If you are attempting to use hardware support and `/dev/random` or `/dev/urandom` failed, OpenSSH will revert to using **ssh-rand-helper** and continue.

2. **The system administrator sees the following messages on the console:**

```
BPXP015I HFS PROGRAM /bin/ssh IS NOT MARKED PROGRAM CONTROLLED.
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR DAEMON (BPX.DAEMON) PROCESSING
```

   A user invoked **ssh** from a user ID that has READ access to BPX.DAEMON. A user ID that is given READ access to BPX.DAEMON should be set up as a protected user ID (for example, with the NOPASSWORD option). Doing so prevents UID(0) users from working in the shell, because they would be able to perform unauthenticated setuids. It appears such a user does have shell access. The system (or security) administrator should double-check the security setup.

3. **I was trying to copy a 6GB file to a remote host using scp. The scp progress meter counted up to 100 percent copied. I received a 'No space left on device' error message but I found out that the file system on the remote host didn't have enough space to begin with. Should scp terminate as soon as the remote file system is full?**

   The server-side **scp** process will not return an out-of-space error until the client has finished transmitting all its data. If you are concerned about running out of space, run a remote command to check the file system space (such as **df** or **zfsadm**) on the remote host before issuing the **scp** command.

4. **When a user logs on via the ssh client, we are getting the following message in the system log: EZZ9297E UNABLE TO ACCESS FILE /etc/resolv.conf. - RC 00101708. The user can still ssh in successfully, but what does this warning mean?**

   The OpenSSH daemon runs with privilege separation enabled by default. During privilege separation, the daemon cleaves itself into two processes, one with privileges and one without. The unprivileged user (the SSHD privilege separation user) handles network traffic and everything not requiring special privileges. This unprivileged process runs in a chroot jail of `/var/empty`. The chroot service changes the root directory from the current one to a new one; in this case, `/var/empty`. The root directory is the starting point for path searches of path names beginning with a slash. At some point, the privilege separation user invokes a TCP/IP system call which requires access to the TCPIP.DATA file. If this file is stored in the UNIX file system as `/etc/resolv.conf`, the privilege separation user will not have access to the file because it is not located off the new root file system of `/var/empty`. The

system administrator should copy `/etc/resolv.conf.` to `/var/empty/etc/resolv.conf.` in order to make this file visible to the privilege separation user.

5. **I am trying to use ssh with public key authentication, but it can't seem to find my keys. What is happening?**

   It is likely that you are running **ssh** from a user that shares a UID. The **ssh** command description in "ssh" on page 85 provides a tip for avoiding problems when running as a user that shares a UID.

6. **When I attempt to start the sshd daemon, I see the following error message, and the sshd daemon does not start.**

   ```
   "FOTS1451 Privilege separation user sshd does not exist"
   ```

   The **sshd** daemon runs with privilege separation enabled by default. Using privilege separation requires that a special user be created. For more information, see "Step for creating the sshd privilege separation user" on page 38.

7. **When I attempt to start the sshd daemon, I see the following error message, and the daemon does not start. "/etc/ssh/sshd_config: EDC5129I No such file or directory. (errno2=0x05620062)"**

   The **sshd** daemon will not start without a configuration file. The default location for this file is `/etc/ssh/sshd_config`. Verify that you have performed all the setup to run the **sshd** daemon. See "Steps for creating or editing configuration files" on page 24 for information about copying the **sshd_config** file.

8. **If I attempt to start the sshd daemon, I see the following error in the syslog: "FOTS1464 Cannot bind any address".**

   Take the following actions:

   a. Verify that port 22 is not reserved in your TCP/IP setup and that port 22 is not in use by another application or another **sshd** daemon. By default, the **sshd** daemon uses port 22. However, the port can be changed by using the **sshd_config** keyword Port.
   b. Verify that the program control attribute is set for the **sshd** daemon.
   c. Verify that the invoking user ID is defined as UID(0) and has READ access to the BPX.DAEMON profile in the FACILITY class.

   For more information about **sshd** daemon setup and startup, see Chapter 5, "For system administrators," on page 21.

9. **When I run an OpenSSH command and receive an error message, I do not see a message number (for example, FOTSnnnn) associated with it.**

   Verify that the _ZOS_OPENSSH_MSGCAT environment variable is unset or set to "openssh.cat" before running the command. For more information, see "Setting up the message catalog for IBM Ported Tools for z/OS: OpenSSH" on page 38. If you have verified that your setup is correct and you are still not seeing message numbers, it could be that the output in question is considered "log" output that might or might not be an error message.

10. **When I run ssh-keyscan, it does not return the host key for a particular host and exits with a 0 (success) return value. I know the host has sshd running. Why aren't I getting any host key output?**

    By default, **ssh-keyscan** returns only protocol version 1 keys. The **sshd** daemon might only be running protocol version 2. Try issuing **ssh-keyscan** again with a protocol version 2 key type.

    Example:

    ```
    ssh-keyscan -t dsa hostname
    ```

11. **When I run ssh-keyscan, I receive the following error: FOTS0414 hostname: exception! What does this mean?**

    This error is often the result when the remote server is down or not running a **sshd** daemon.

12. **When I invoke ssh, it seems to have poor performance. In particular, if I run in verbose mode (ssh -vvv), it appears to hang on the following line: debug1: ssh_connect: needpriv 0**

    ssh performs some DNS lookups. If the DNS server is down, some operations may take a while to time-out. Verify that DNS is configured properly. Check that the servers in the DNS resolution files (for example, `/etc/resolv.conf`) are working.

13. **When I use the ~# escape sequence to display forwarded connections, not all of them are displayed.**

    Check if you have nested **ssh** clients. For nested **ssh** clients, escape characters are captured and processed by parent **ssh** processes first. To allow an escape sequence to pass through to a child **ssh** client, you can escape the escape character; for example, "~~".

14. **My sftp session hangs when I try to use subcommand 'ls', 'get' or 'put'.**

    You probably have a MTU fragmentation problem. Reduce the TCP/IP MTU (maximum transmission unit) by using the **ifconfig** command.

    Example:

    ```
    ifconfig enth0 mtu 1500
    ```

    Also, specifying a smaller buffer size (the default is 32768) on the **sftp** command line can be a workaround.

    Example:

    ```
    sftp -B 1024 user@host
    ```

15. **scp between two remote hosts doesn't work for me. I specified 'ForwardAgent yes' in my own configuration file and used '-F usr_config_file' to invoke it.**

    When doing **scp** between two remote hosts, you need to specify 'ForwardAgent yes' in the **ssh** global configuration file `/etc/ssh/ssh_config` or the **ssh** default per-user configuration file `~/.ssh/config`. The command-line option **-F** *usr_config_file* does not get passed to the remote host. **scp** only passes options **-v, -r** or **-p** to the remote host regardless of what you specify on the command line.

16. **When I run sftp with protocol version 1 from z/OS to AIX, I keep getting "FOTS0841 Connection closed"**

    Due to a limitation of SECSH protocol and how OpenSSH uses channels, **sftp** for protocol version 1 is only supported between z/OS hosts.

17. **My session hangs part way through logging on when I try to do 'sftp -s sftp_server_path usr@host' between z/OS and Linux. I use protocol version 2.**

    User-defined subsystems (those specified with the **-s** option) are only supported between z/OS hosts. This is due to a limitation of the SECSH protocol with regards to EBCDIC platforms.

18. **When I use ssh with the -s option to utilize a subsystem, my session hangs while logging on. I am using protocol version 2.**

    User-defined subsystems (those specified with the **-s** option) are only supported between z/OS hosts. This is due to a limitation of the SECSH protocol with regards to EBCDIC platforms.

19. **When I attempt to start ssh, I get the error message "FOTS0944 buffer_get_bignum_ret: input buffer too small"**.

    Your public key or private key file might be corrupted. Regenerate your keys and try again.

20. **When I attempt to copy a file using scp or sftp, after user authentication succeeds, the command fails and exits with a nonzero (failure) return code. I also saw some output from a sshrc file when using scp.**

    This error is often seen when the user has `/etc/ssh/sshrc` or `~/.ssh/rc` on the remote host that is generating output to stdout. Make sure that both `/etc/ssh/sshrc` and `~/.ssh/rc` do not send output to stdout when either **scp** or **sftp** is used. Instead, the output should be written to stderr. (Output generated from the **sshrc** file is displayed for **scp** but not for **sftp**.)

21. **When I ssh to a remote host using public key or password authentication, I never get a chance to enter the passphrase/password, instead receiving the following error: "FOTS1346 Permission denied, please try again". This causes user authentication to fail. The ssh client then eventually fails with the error: "FOTS1373 Permission denied (publickey,password,keyboard-interactive)".**

    Verify that you are not trying to use **ssh** while switched to another user ID. In other words, did you issue **ssh** after the **su** command? The original controlling terminal (displayed by the **tty** command) is owned by the user ID originally logged in. Your target user might not have permission to read from it.

22. **I attempt to start sftp but I receive error message "FOTS0843 Received message too long xxxx" where xxxx is the length of message.**

    Possibly, an sftp packet was corrupted by TCP/IP RESOLVER trace output written to stdout. To check whether RESOLVER trace output is being sent to stdout, issue the following shell command on both the local host and the remote host:

    ```
    netstat -S
    ```

    If you see messages about RESOLVER trace initialization in the output of the netstat command, then it means the RESOLVER trace output is written to stdout on the system that you issued the netstat command. You can redirect RESOLVER trace output to avoid conflicts with **sftp** by issuing the following:

    ```
    export RESOLVER_TRACE=STDERR
    ```

    If the RESOLVER trace output is enabled on the remote host (the system running the daemon), the daemon will need to be restarted with the new environment.

23. **The sshd daemon fails to start and the stderr file contains: "The signal SIGHUP was received."**

    You have come across a process race condition. You will need to do some setup tasks as described in "Using BPXBATCH" on page 40.

24. **Sometimes when I run the ssh command on z/OS, I get the following SIGINT messages:**

    ```
    /u/user> ssh jim@remotehost
    CEE5206S THE SIGNAL SIGINT WAS RECEIVED.
    ```

    The command completes and I am able to log into the remote host.

    The OpenSSH base distribution added functionality to the random number generator, **ssh-rand-helper**. Specifically, if an invoked UNIX command (from the `/etc/ssh/ssh_prng_cmds` file) is taking too long, it will be killed by a SIGINT signal. You might see this message if your system is heavily loaded. In previous versions of OpenSSH, the process was not killed. Instead,

processing continued to the next UNIX command in the file. You might see this message displayed from any of the OpenSSH utilities, not just the **ssh** client.

The system administrator might also see the following message on the console:

```
IEF450I JOBNAME *OMVSEX - ABEND=SEC6 U0000 REASON=0000FF02
```

The console message results when **ssh-rand-helper** kills the UNIX command listed in `/etc/ssh/ssh_prng_cmds` before the kernel is able to initialize the child process for the command. Again, you might see the console message if your system is heavily loaded.

Both messages can be eliminated by having Integrated Cryptographic Service Facility (ICSF) available because OpenSSH uses hardware support (`/dev/random` or `/dev/urandom`) to generate random numbers instead of using **ssh-rand-helper**. For more information about using hardware support, see "ssh-rand-helper" on page 115.

If ICSF is not available, then the **ssh-rand-helper** timeout value can be increased in order to eliminate both messages. For more information about the timeout value, see "ssh-rand-helper — Gather random numbers for OpenSSH" on page 115.

25. **When I use the stty command in a shell profile to set the terminal options for my interactive z/OS OpenSSH session, I see the following error message: "stty: FSUMB039 error setting termios attributes: EDC5139I Operation not permitted"**.

    The extended packet mode terminal option (PKTXTND in termios.h) setting was changed under APAR OA12576 in the previous release. The option is now turned on. Therefore, using the **stty** command to turn off the PKTXTND option within an interactive z/OS OpenSSH session will fail. Your **stty** command needs to be updated to leave the PKTXTND option unchanged (that is, turned on).

Setting up syslogd to debug sshd

Setting up the syslog daemon (**syslogd**) can help to debug **sshd** problems. For more information about configuring **syslogd**, see *z/OS Communications Server: IP Configuration Guide*.

Steps for setting up syslogd to debug sshd

Before you begin: You need to have superuser authority in order to start the **syslogd** daemon.

Perform the following steps to set up **syslogd** to debug **sshd**.

1. Create the **syslogd** configuration file `/etc/syslog.conf`.
   a. Create directory `/tmp/syslogd`.

      ```
      mkdir /tmp/syslogd
      ```

   b. Add a configuration statement in the `syslogd.conf` file.

      Example:

      ```
      echo "daemon.debug   /tmp/syslogd/server.logfile" >> /etc/syslog.conf
      ```

      Result: Writes debug messages with facility daemon to `/tmp/syslogd/server.logfile`.

   c. Set the permission bits.

      ```
      chmod 644 /etc/syslog.conf
      ```

d. Create the log file.

```
touch  /tmp/syslogd/server.logfile
```

---

2. Start **syslogd**

```
/usr/sbin/syslogd -f /etc/syslog.conf &
```

---

3. In the **sshd_config** configuration file, add the SyslogFacility and LogLevel keywords. The default SyslogFacility is AUTH. The default LogLevel is INFO.

   Example:

```
SyslogFacility DAEMON
LogLevel    DEBUG3
```

---

4. To force **syslogd** or **sshd** to reread its configuration files and activate any modified parameters without stopping, issue:

```
kill -s SIGHUP PID
```

   where PID is the process ID of **syslogd** or **sshd**.

When you are done, you have set up **syslogd**.

Chapter 14. OpenSSH vulnerabilities

List of vulnerabilities reported against OpenSSH applications

Table 36 lists vulnerabilities reported by Carnegie Mellon University Software Engineering Institute's CERT Coordination Center (CERT/CC) and by Common Vulnerabilities and Exposures (CVE), which is sponsored by the National Cyber Security Division at the U.S. Department of Homeland Security. The listed vulnerabilities are against OpenSSH. The version of OpenSSH used is 5.0p1.

Table 36. List of vulnerabilities reported against OpenSSH applications

CERT/CVE	Date	Public name description	Is OpenSSH vulnerable?
CVE-2004-1653	08/31/2004	OpenSSH could allow remote authenticated users to perform a port bounce, when configured with an anonymous access program	No, if you retain the default value of "no" for the **sshd_config** AllowTcpForwarding keyword or if you do not configure OpenSSH with an anonymous access program such as AnonCVS.
CVE-2007-2243	04/25/2007	OpenSSH, when ChallengeResponseAuthentication is enabled, allows remote attackers to determine the existence of user accounts	No. OpenSSH does not support challenge-response authentication.
CVE-2007-2768	05/21/2007	OpenSSH, when using OPIE (One-Time Passwords in Everything) for PAM, allows remote attackers to determine the existence of certain user accounts.	No. OpenSSH does not support PAM.
CVE-2008-3259	07/22/2008	OpenSSH sets the SO_REUSEADDR socket option when the X11UseLocalhost configuration setting is disabled, which allows local users on some platforms to hijack the X11 forwarding port via a bind to a single IP address.	No. OpenSSH on z/OS has applied the patch (fix) for this security vulnerability.
CVE-2008-5161	11/19/2008	Error handling in the SSH protocol when using a block cipher algorithm in Cipher Block Chaining (CBC) mode, makes it easier for remote attackers to recover certain plaintext data from an arbitrary block of ciphertext in an SSH session via unknown vectors.	No, if you do not use the CBC mode ciphers. If the CBC mode ciphers are used, OpenSSH has applied the patch (fix) that contains countermeasures to mitigate the security vulnerability.

For more information, see the US-CERT Vulnerability Notes Database at http://www.kb.cert.org/vuls and the National Vulnerability Database at http://nvd.nist.gov/nvd.cfm.

List of vulnerabilities reported against zlib

zlib is a data compression library used by OpenSSH. Currently, there are no reported vulnerabilities against zlib version 1.2.3.

List of vulnerabilities reported against OpenSSL

Table 37 lists vulnerabilities reported by CERT/CC and by CVE against OpenSSL. OpenSSL provides cryptographic library functions used by OpenSSH. The version of OpenSSL used is 0.9.8k.

Table 37. List of vulnerabilities reported against OpenSSL applications

CERT/CVE	Date	Public name description	Is OpenSSH vulnerable?
CVE-2009-1377	05/19/2009	The dtls1_buffer_record function in ssl/d1_pkt.c allows remote attackers to cause a denial of service (memory consumption) via a large series of "future epoch" DTLS records that are buffered in a queue.	No. OpenSSH on z/OS does not use the vulnerable code.
CVE-2009-1378	05/19/2009	Multiple memory leaks in the dtls1_process_out_of_seq_message function in ssl/d1_both.c allows remote attackers to cause a denial of service (memory consumption) via DTLS records that (1) are duplicates or (2) have sequence numbers much greater than current sequence numbers.	No. OpenSSH on z/OS does not use the vulnerable code.
CVE-2009-1379	05/19/2009	Use-after-free vulnerability in the dtls1_retrieve_buffered_fragment function in ssl/d1_both.c allows remote attackers to cause a denial of service (openssl s_client crash) and possibly have unspecified other impact via a DTLS packet.	No. This vulnerability affects OpenSSL 1.0.0 Beta 2. OpenSSH on z/OS utilizes OpenSSL 0.9.8k.
CVE-2009-1387	06/04/2009	The dtls1_retrieve_buffered_fragment function in ssl/d1_both.c allows remote attackers to cause a denial of service (NULL pointer dereference and daemon crash) via an out-of-sequence DTLS handshake message, related to a "fragment bug".	No. OpenSSH on z/OS does not use the vulnerable code.
CVE-2009-3245	03/05/2010	OpenSSL does not check for a NULL return value from bn_wexpand function calls in (1) crypto/bn/bn_div.c, (2) crypto/bn/bn_gf2m.c, (3) crypto/ec/ec2_smpl.c, and (4) engines/e_ubsec.c, which has unspecified impact and context-dependent attack vectors.	No. OpenSSH on z/OS does not use the vulnerable code.
CVE-2009-3555	11/09/2009	The TLS and SSL protocols do not properly associate renegotiation handshakes with an existing connection.	No. OpenSSH on z/OS does not use the TLS or SSL protocols for handshake renegotiation.
CVE-2009-4355	01/14/2010	Memory leak in the zlib_stateful_finish function in crypto/comp/c_zlib.c allows remote attackers to cause a denial of service (memory consumption) via vectors that trigger incorrect calls to the CRYPTO_cleanup_all_ex_data function.	No. OpenSSH on z/OS does not use the vulnerable code.

Table 37. List of vulnerabilities reported against OpenSSL applications (continued)

CERT/CVE	Date	Public name description	Is OpenSSH vulnerable?
CVE-2010-0433	03/05/2010	The kssl_keytab_is_available function in ssl/kssl.c does not check a certain return value, which allows remote attackers to cause a denial of service (NULL pointer dereference and daemon crash) via SSL cipher negotiation.	No. OpenSSH on z/OS does not use the vulnerable code.
CVE-2010-0740	03/26/2010	The ssl3_get_record function in ssl/s3_pkt.c allows remote attackers to cause a denial of service (crash) via a malformed record in a TLS connection that triggers a NULL pointer dereference, related to the minor version number.	No. OpenSSH on z/OS does not use the vulnerable code.

List of past vulnerabilities that affected IBM Ported Tools for z/OS: OpenSSH in Version 1 Release 1

These past vulnerabilities do not affect Version 1 Release 2 of IBM Ported Tools for z/OS: OpenSSH. They are listed for historical purposes only. For more information, see the US-CERT Vulnerability Notes Database at http://www.kb.cert.org/vuls and the National Vulnerability Database at http://nvd.nist.gov/nvd.cfm.

OpenSSH

VU#389665
CVE-2002-1357

VU#978316
CVE-2003-0386

VU#333628
CVE-2003-0693

VU#602204
CVE-2003-0786

VU#209807
CVE-2003-0787

CVE-2004-0175

CVE-2005-2666

CVE-2005-2797

CVE-2005-2798

CVE-2006-0225

VU#787448
CVE-2006-4924

CVE-2006-4925

VU#851340
CVE-2006-5051

CVE-2006-5052

CVE-2006-5794

CVE-2007-4752

CVE-2008-1483

CVE-2008-1657

zlib

VU#368819
CVE-2002-0059

VU#142121
CVE-2003-0107

VU#238678
CVE-2004-0797

CVE-2005-1849

VU#680620
CVE-2005-2096

OpenSSL

VU#888801
CVE-2003-0131

VU#997481
CVE-2003-0147

VU#255484
CVE-2003-0543

VU#380864
CVE-2003-0544

VU#935264
CVE-2003-0545

VU#412478
CVE-2003-0851

VU#288574
CVE-2004-0079

VU#465542
CVE-2004-0081

VU#484726
CVE-2004-0112

CVE-2005-1797

CVE-2005-2946

CVE-2005-2969

VU#247744
CVE-2006-2937

VU#423396
CVE-2006-2940

VU#547300
CVE-2006-3738

VU#845620
CVE-2006-4339

VU#386964
CVE-2006-4343

VU#724968
CVE-2007-3108

CVE-2007-4995

CVE-2007-5135

VU#661475
CVE-2008-0891

VU#520586
CVE-2008-1672

CVE-2008-1678

CVE-2008-5077

CVE-2009-0590

CVE-2009-0591

CVE-2009-0789

CVE-2009-1386

Chapter 15. OpenSSH messages

FOTS0101 unknown key type *type*

Explanation: You specified an option that is not valid for this command.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a list of options.

FOTS0102 bad key type

Explanation: Incorrect key type was passed.

System action: Command ends.

User response: Verify that the key file entered is valid.

FOTS0103 load failed

Explanation: Either the specified file is not the correct type or the passphrase was incorrect.

System action: Command ends.

User response: Check the file, the specified passphrase, and try the command again.

FOTS0104 fgets failed

Explanation: **ssh–keygen** could not read the answer to the prompt.

System action: Command ends.

User response: Try reissuing **ssh–keygen** with options for input instead of prompts. Check *IBM Ported Tools for z/OS User's Guide* for a list of options.

FOTS0105 key_to_blob failed

Explanation: **ssh–keygen** could not convert the key from OpenSSH format.

System action: Command ends.

User response: Check that the key specified is OpenSSH format.

FOTS0106 input line too long.

Explanation: **ssh–keygen** could not convert the key. Data in the key file had a line that was too long.

System action: Command ends.

User response: Check that you specified the correct key file, and try again.

FOTS0107 uudecode failed.

Explanation: **ssh–keygen** could not convert the key because uudecode() failed.

System action: Command ends.

User response: Check that you specified the correct key file, and try again.

FOTS0108 decode blob failed.

Explanation: **ssh–keygen** could not convert the key.

System action: Command ends.

User response: Check that you specified the correct key file, and try again.

FOTS0109 key_write failed

Explanation: The key information could not be written to either stdout or file.

System action: Command ends.

User response: If using options to create or change the key file, check that there is enough space to create a key file.

FOTS0110 *filename* **is not a public key file**

Explanation: The command expected the file to be a public key and it is not.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for the options description.

FOTS0111 Bad passphrase.

Explanation: The key file could not be loaded. Either the file given is not the correct format or the passphrase is not correct.

System action: Command ends.

User response: Check the file and the passphrase, and try again.

FOTS0112 Passphrases do not match. Try again.

Explanation: The two passphrases given were not the same.

System action: Command ends.

User response: You need to specify the same passphrase twice.

FOTS0113 Saving the key failed: *filename*.

Explanation: The key file could not be saved.

System action: Command ends.

User response: Verify that you have correct permissions to create the key file.

FOTS0114 Could not create directory '*directory*'.

Explanation: The mkdir() failed and could not create the *directory* directory.

System action: Command ends.

User response: Check that you have correct permissions to create directory.

FOTS0115 Comments are only supported for RSA1 keys.

Explanation: Comments can only be changed for RSA1 key types.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a list of options and descriptions.

FOTS0116 Key now has comment '*string*'

Explanation: Informational message when comment is changed.

System action: Command continues.

User response: None.

FOTS0117 Enter new comment:

Explanation: This is a prompt for specifying a new comment.

System action: Command waiting for input.

User response: Specify the new comment.

FOTS0118 Could not save your public key in *filename*

Explanation: Creation of the public file failed.

System action: Command ends.

User response: Check that you have correct permissions to create the file.

FOTS0119 fdopen *filename* failed

Explanation: The system call fdopen() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0120 key_generate failed

Explanation: Could not generate the private key.

System action: Command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0121 You don't exist, go away!

Explanation: The getpwuid() system call failed. This may happen when there are multiple users with the same UID and one of them does not have the group defined in the OMVS segment or the default group does not have OMVS segment.

System action: Command ends.

User response: Check the users for the group and the default group.

FOTS0122 Bits has bad value.

Explanation: Allowed range is 768 to 32768 bits.

System action: Command ends.

User response: Change the bits value and reissue the command.

FOTS0123 Too many arguments.

Explanation: You specified arguments that are mutually exclusive.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a list of options.

FOTS0124 Can only have one of -p and -c.

Explanation: You cannot change both the passphrase and the comment in the same command. You have to change them one at a time.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a list of options.

FOTS0125 You must specify a key type (-t).

Explanation: You need to specify the key type when generating a key file. Option -t type and -d specify the key format.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's*

Guide for the correct format.

FOTS0126 **buffer_get_bignum_bits: input buffer too small: need** *need_bits* **have** *have_bits*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0127 **bad magic 0x***magic_value* **!= 0x***expected_value*

Explanation: Unexpected value in private key.

System action: Command ends.

User response: Check that you specified the correct key file, and try again.

FOTS0128 **unsupported cipher** *cipher*

Explanation: The specified cipher for the key is not supported.

System action: Command ends.

User response: Check that you specified the correct key file, verify that the cipher used to create the key is supported, and then try again.

FOTS0129 **line** *number* **too long:** *line***..."**

Explanation: **ssh–keygen** could not convert the key. Data in the key file had a line that was too long.

System action: Command ends.

User response: Check that you specified the correct key file, and try again.

FOTS0130 **do_convert_private_ssh2_from_blob: remaining bytes in key blob** *rlen*

Explanation: **ssh–keygen** could not convert the key.

System action: Command continues.

User response: Check that you specified the correct key file, and try again.

FOTS0131 **strtol failed:**

Explanation: A call to strtol() failed. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0132 **version 1 keys are not supported**

Explanation: The -e option cannot be used with RSA keys for use by SSH protocol version 1.

System action: The program ends.

User response: Specify a RSA key for use by SSH protocol version 2 or a DSA key.

System programmer response: Not applicable

FOTS0133 **Primality trials has bad value.**

Explanation: Number of primality trials must be an integer greater than or equal to 4.

System action: The command ends.

User response: Select an integral value greater than or equal to 4.

System programmer response: Not applicable

FOTS0134 **Desired generator has bad value.**

Explanation: Generator value must be greater than or equal to 1.

System action: The command ends.

User response: Select a generator value greater than or equal to 1.

System programmer response: Not applicable

FOTS0135 **Minimum primality trials is** *TRIAL_MINIMUM*

Explanation: The number of trials specified must be greater than or equal to *TRIAL_MINIMUM*.

System action: The command ends.

User response: Select a trials value greater than or equal to *TRIAL_MINIMUM*.

System programmer response: Not applicable

FOTS0136 **Invalid memory amount (min** *min_memory***, max** *max_memory***)**

Explanation: The memory amount must be greater than or equal to *min_memory* and less than or equal to *max_memory*.

System action: The command ends.

User response: Select a memory value greater than or equal to *min_memory* and less than or equal to *max_memory*.

System programmer response: Not applicable

FOTS0137 Invalid start point.

Explanation: A call to OpenSSL function BN_hex2bn() failed for the specified start point.

System action: The program ends.

User response: Make sure the specified start point is a string which begins with one or more valid hexadecimal digits. If the specified string is valid and the problem persists then contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0138 Couldn't open modulus candidate file "*filename*": *error_message*

Explanation: A call to fopen() failed on file *filename* The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0139 modulus candidate generation failed

Explanation: Internal error.

System action: The command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0140 Couldn't open moduli file "*filename*": *error_message*

Explanation: A call to fopen() failed on file *filename*. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0141 modulus screening failed

Explanation: Internal error.

System action: The command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0142 Memory option has bad value.

Explanation: The value specified for the memory option must be an integer greater than 7 and less than 128.

System action: The command ends.

User response: Select an integer value greater than 7 and less than 128.

System programmer response: Not applicable

FOTS0143 buffer_get_bignum_bits: BN_bin2bn failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0144 hash_host failed

Explanation: Internal error. Unable to hash host name information.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0145 Specified known hosts path too long

Explanation: The known_hosts file path name is too long.

System action: The program ends.

User response: Verify that the path name of the known_hosts file is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0146 fopen: *error_message*

Explanation: The fopen() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0147 known_hosts path too long

Explanation: The known_hosts file path name is too long.

System action: The program ends.

User response: Verify that the path name of the known_hosts file is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0148 mkstemp: *error_message*

Explanation: The mkstemp() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0149 fdopen: *error_message*

Explanation: The fdopen() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0150 line *line_number* **missing key:** *line_in_error***...**

Explanation: Line *line_number* in the known_hosts file is missing key information.

System action: The program continues.

User response: Verify that a valid known_hosts file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0151 line *line_number* **invalid key:** *line_in_error***...**

Explanation: Line *line_number* in the known_hosts file contains an invalid key.

System action: The program continues.

User response: Verify that a valid known_hosts file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0152 line *line_number***: invalid hashed name:** *line_in_error***...**

Explanation: Line *line_number* in the known_hosts file contains a hashed host name that is not valid.

System action: The program continues.

User response: Verify that a valid known_hosts file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0153 Warning: ignoring host name with metacharacters: *host_name*

Explanation: Skipped hashing host name *host_name* with metacharacters.

System action: The program continues.

User response: If you expected all host names to be hashed, verify that a valid known_hosts file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0154 *filename* **is not a valid known_hosts file.**

Explanation: An error occurred while processing the known_hosts file *filename*.

System action: The program ends.

User response: Verify that a valid known_hosts file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0155 Not replacing existing known_hosts file because of errors

Explanation: The existing known_hosts file was not replaced because an error occurred while processing the file.

System action: The program ends.

User response: Verify that a valid known_hosts file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0156 unlink *filename*: *error_message*

Explanation: The unlink() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0157 link *filename1* **to** *filename2*: *error_message*

Explanation: The link() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0158 rename "*filename1***" to "***filename2***":** *error_message*

Explanation: The rename() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0159 Identity filename too long

Explanation: The specified identity filename is too long.

System action: The program ends.

User response: Specify a valid identity filename, and try the request again.

FOTS0160 Output filename too long

Explanation: The specified output filename is too long.

System action: The program ends.

User response: Specify a valid output filename, and try the request again.

FOTS0161 no keys found.

Explanation: No keys were found in the key file.

System action: The program ends.

User response: Verify that a valid key file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0162 no support for smartcards.

Explanation: **ssh–keygen** on z/OS does not support smart cards.

System action: The program ends.

User response: Do not specify **ssh–keygen** smart card options. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0163 DSA keys must be 1024 bits

Explanation: The **ssh–keygen** bits value for the DSA key is not 1024.

System action: The program ends.

User response: Correct the **ssh–keygen** bits value, and try the request again.

FOTS0164 ungetc: *error_message*

Explanation: The ungetc() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0165 *filename* **updated.**

Explanation: The known_hosts file *filename* was updated.

System action: The program continues.

User response: None.

FOTS0166 Original contents retained as *filename*

Explanation: The original contents of the known_hosts file is retained in file *filename*.

System action: The program continues.

User response: None.

FOTS0167 WARNING: *filename* contains unhashed entries

Explanation: The known_hosts file *filename* contains unhashed host names. The file should be deleted to ensure privacy.

System action: The program continues.

User response: Delete file *filename* to ensure privacy of the host names.

FOTS0169 Entering new comment failed: *filename*.

Explanation: Failed to enter new comment for key file *filename*.

System action: The program ends.

User response: Verify that a valid key file is specified, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0170 Invalid number of trials: *number_of_trials* (*error_message*)

Explanation: The specified **ssh–keygen** number of trials value is not valid. The error is displayed with the message.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid **ssh–keygen** number of trials values, and try the request again.

FOTS0171 Memory limit is *error message*: *memory_limit*

Explanation: The specified **ssh–keygen** memory limit value is not valid. The error is displayed with the message.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid **ssh–keygen** memory limit values, and try the request again.

FOTS0172 Bits has bad value *bits* (*error_message*)

Explanation: The specified **ssh–keygen** bits value is not valid. The error is displayed with the message.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid **ssh–keygen** bits values, and try the request again.

FOTS0173 Desired generator has bad value: *generator* (*error_message*)

Explanation: The specified **ssh–keygen** generator value is not valid. The error is displayed with the message.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid **ssh–keygen** generator values, and try the request again.

FOTS0201 *variable* not set, cannot kill agent

Explanation: *variable* environment variable was not set so **ssh–agent** could not get the PID of the agent to kill

System action: Command ends.

User response: Set the *variable* environment variable to the correct agent pid.

FOTS0202 *variable*="*value*",which is not a good PID

Explanation: The *variable* environment variable does not contain the correct pid so the agent could not be killed.

System action: Command ends.

User response: Check the *variable* environment variable and its value.

FOTS0203 internal error, bad protocol version *version*

Explanation: **ssh–agent** supports version 1 and 2. The displayed version is not supported.

System action: Command ends.

User response: Contact your system administrator to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0204 process_remove_identity: internal error: tab->nentries *number*

Explanation: Failure occurred during internal processing of removing keys.

System action: Command ends.

User response: Contact your system administrator to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0205 select: *message*

Explanation: select() system call failed

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0206 Unknown message *number*

Explanation: **ssh–agent** could not process the given message.

System action: Command ends.

User response: Contact your system administrator to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0207 fcntl O_NONBLOCK: *message*

Explanation: fcnt() system call failed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0208 accept from AUTH_SOCKET: *message*

Explanation: accept() system call failed. could not get correct socket number

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0209 getpeereid *id* **failed:** *message*

Explanation: getpeereid fails for the given socket.

System action: The socket gets closed and command continues.

User response: Check the system error message which follows this message.

FOTS0210 uid mismatch: peer euid *id* **!= uid** *uid*

Explanation: ssh–agent sockets are owned by the uid which created it and can only be used by that uid and superuser.

System action: Command continues.

User response: Check that you are using the correct uid and SSH_AUTH_SOCK environment variable has correct value.

FOTS0211 kill

Explanation: kill system call failed and could not kill the agent.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0212 mkdtemp: private socket dir

Explanation: Could not create the private directory for agent socket.

System action: Command ends.

User response: Check the system error message which follows this message.

FOTS0213 socket

Explanation: Could not create socket because socket system call failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0214 bind

Explanation: bind system call failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0215 listen

Explanation: listen system call failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0216 fork

Explanation: fork system call failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0217 setenv

Explanation: setenv system call failed and **ssh–agent** could not set either SSH_AUTH_SOCK or SSH_AGENT_PID variables.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0218 setsid: *message*

Explanation: setsid system call failed

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0219 setrlimit RLIMIT_CORE: *string*

Explanation: setrlimit system call failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0220 process_authentication_challenge1: BN_new failed

Explanation: The BN_new function failed.

System action: Command ends.

User response: Contact your system administrator to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0221 Unknown socket type *number*

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0222 Unknown type *number*

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0231 process_add_identity: RSA_blinding_on failed

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0232 *variable*=**"***value***", which is not a good PID:** *error_message*

Explanation: The *variable* environment variable does not contain the correct pid so the agent could not be killed.

System action: The program ends.

User response: Check the *variable* environment variable and its value and try the request again.

FOTS0233 process_authentication_challenge: bad challenge length *length*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0234 Warning: identity keysize mismatch: actual *keysize1***, announced** *keysize2*

Explanation: Possible RSA key problem encountered while removing identity from the agent.

System action: The program continues.

User response: Verify that the RSA key is valid and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0301 Bad key file *filename*

Explanation: The public key of the specified identity could not be loaded.

System action: Command continues to the next file (if any).

User response: Make sure the public key exists in the same directory as the pathname of the identity.

FOTS0302 Failed to remove all identities.

Explanation: One or more version 1 identities could not be removed from the ssh-agent when trying to remove all.

System action: Command ends.

User response: Check what identities are still present in the ssh-agent. Contact system programmer.

FOTS0303 Could not remove identity: *filename*

Explanation: **ssh–agent** returned a bad code when removal was attempted.

System action: Command continues to next identity (if any).

User response: Contact system programmer.

FOTS0304 Could not add identity: *filename*

Explanation: The specified identity could not be added to the ssh-agent.

System action: Command continues to next file (if any).

User response: Contact system programmer.

FOTS0305 key_write failed

Explanation: The key parameter could not be written to the stdout.

System action: Command continues.

User response: Not applicable

FOTS0306 Passwords do not match.

Explanation: When prompted twice for the password, the passwords must match.

System action: Command ends.

User response: Retry command giving the same password twice.

FOTS0307 Failed to (un)lock agent.

Explanation: The ssh-agent could not be either locked or unlocked.

System action: Command ends.

User response: If unlocking, check that correct password was given. When unlocking, check that the same password was given twice.

FOTS0308 Could not open a connection to your authentication agent.

Explanation: **ssh–add** needs **ssh–agent** to be running to execute.

System action: Command ends.

User response: Check that you have **ssh–agent** running and the SSH_AGENT_PID and SSH_AUTH_SOCK environment variables hold the agent data and are exported.

FOTS0309 Invalid lifetime

Explanation: The format of the -t argument was incorrect and the lifetime could not be set.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a list of options.

FOTS0310 Smartcards are not supported

Explanation: You tried to use -s or -e option which is not supported.

System action: Command ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a list of options.

FOTS0311 No user found with uid *uid*

Explanation: The getpwuid() system call failed. This may happen when there are multiple users with the same uid and one of them does not have the group defined in the omvs segment or the default group does not have omvs segment.

System action: Command ends.

User response: Check the users for the given uid for the group and the default group.

FOTS0327 *identity_file* **:** *message*

Explanation: A call to stat() failed on file *identity_file*. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0401 Impossible! dispatch_run() returned!

Explanation: Call to dispatch_run returned when it should not have.

System action: Command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0402 Bad port '*port_num***'**

Explanation: The specified port number is not valid.

System action: Command ends.

User response: Specify a valid port number.

FOTS0403 Bad timeout '*time***'**

Explanation: The specified timeout value is not valid.

System action: Command ends.

User response: Specify a valid timeout value.

FOTS0404 *hostname***: invalid packet type**

Explanation: Packet received from host was not in the proper format.

System action: Command continues.

User response: Verify connections. If problem persists contact your system administrator to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0405 getaddrinfo *hostname***:** *message*

Explanation: A call to getaddrinfo() failed. The system error is displayed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0406 socket: *message*

Explanation: A call to socket() failed. The system error is displayed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0407 F_SETFL: *error_message*

Explanation: fcntl() system call failed.

System action: Command ends

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0408 connect ('*hostname***'):** *message*

Explanation: A call to connect() failed. The system error is displayed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0409 read ('*hostname***'):** *message*

Explanation: Could not read from socket because the read system call failed. The system error is displayed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error.

If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0410 *hostname*: **Connection closed by remote host.**

Explanation: The remote host has closed the connection.

System action: Command continues.

User response: Contact the remote host sysadmin for further assistance.

FOTS0411 *hostname*: **bad greeting**

Explanation: The greeting received from the server is not in the proper format.

System action: Command continues.

User response: Contact the remote host sysadmin for further assistance.

FOTS0412 **write ('***hostname***'):** *message*

Explanation: Could not write to the socket because the write system call failed. The system error is displayed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0414 *hostname*: **exception!**

Explanation: There is an exception for the socket associated with the indicated hostname. This error is often the result when the remote server is down or not running ssh.

System action: Command continues.

User response: Contact the remote host sysadmin for further assistance.

FOTS0415 **conalloc: fdno** *number* **too high**

Explanation: The file descriptor value exceeds the maximum for the system.

System action: Command ends.

User response: Contact the system programmer for further assistance.

System programmer response: Verify system file descriptor settings. If problem cannot be resolved then follow local procedures for reporting problems to IBM.

FOTS0416 **conalloc: attempt to reuse fdno** *number*

Explanation: The program is attempting to allocate a file descriptor that is already in use.

System action: Command ends.

User response: Contact the system programmer for further assistance.

System programmer response: Verify system file descriptor settings. If problem cannot be resolved then follow local procedures for reporting problems to IBM.

FOTS0417 **confree: attempt to free bad fdno** *number*

Explanation: The program attempted to free a connection that did not exist.

System action: Command ends.

User response: Contact the system programmer for further assistance.

System programmer response: Verify system file descriptor settings. If problem cannot be resolved then follow local procedures for reporting problems to IBM.

FOTS0418 **conread: invalid status** *status*

Explanation: The connection status value is invalid.

System action: Command ends.

User response: Verify the status of hosts being scanned.

FOTS0419 **Too high debugging level.**

Explanation: The specified debugging level exceeds the maximum value of 3.

System action: Command ends.

User response: Specify a debugging level of 3 or less.

FOTS0420 **unknown key type** *keytype*

Explanation: The specified key type is not a valid key type.

System action: Command ends.

User response: Specify a valid key type.

FOTS0421 *progname*: **fdlim_get: bad value**

Explanation: The number of file descriptors available to the process is less than zero.

System action: Command ends.

User response: Contact the system administrator for further assistance.

System programmer response: Verify system file descriptor settings. If problem cannot be resolved then

follow local procedures for reporting problems to IBM.

FOTS0422 *progname*: **not enough file descriptors**

Explanation: The number of file descriptors available to the process for use for connections is zero or less.

System action: Command ends.

User response: Contact the system administrator for further assistance.

System programmer response: Verify system file descriptor settings. If problem cannot be resolved then follow local procedures for reporting problems to IBM.

FOTS0424 *function*: **set_nonblock(***socket***)**

Explanation: **ssh–keyscan** failed to set the connection socket *socket* to non–blocking. The failure occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0425 host_hash failed

Explanation: Failed to hash the hostnames and addresses.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0426 snprintf: buffer too small

Explanation: Failed to set up the connection because an internal buffer was too small.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0501 *progname*: *resource_name* **must be boolean, not** *buf*.

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0502 *progname*: *resource_name* **must be an integer, not** *buf*.

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0503 *progname*: *resource_name* **must be a float, not** *buf*.

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0504 *progname*: **can't parse color** *color*

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0505 *progname*: **couldn't allocate color** *color*

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0506 *appName*[*pid*]: **Aaahhh! I ran out of memory at line** *line*.

Explanation: Out of memory.

System action: Command ends.

User response: Free more system resources and reissue the command.

FOTS0507 *appName*[*pid*]: **invalid value '***string_resource***' for** *instanceName*.

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0508 *appName*[*pid*]**: performGrab: invalid grab type (***grabType***).**

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0509 *appName*[*pid*]**: performGrab: null grab type name.**

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0510 *appName*[*pid*]**: Could not grab** *grabTypeName* **(***reason***)**

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0511 *appName*[*pid*]**: *Yawn*...timed out after** *time* **seconds.**

Explanation: Timed out waiting for user response.

System action: Command ends.

User response: Respond to prompt prior to timeout.

FOTS0512 *appName*[*pid*]**: getrlimit failed (***system error***)**

Explanation: getrlimit() system call failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0513 *appName*[*pid*]**: setrlimit failed (***system error***)**

Explanation: setrlimit() system call failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0514 *appName*[*pid*]**: This should not happen.**

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0701 process_read: seek failed

Explanation: System call lseek() failed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0702 process_write: seek failed

Explanation: System call lseek() failed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0703 process_write: write failed

Explanation: System call write() failed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0704 bad message

Explanation: Internal error.

System action: Command ends.

User response: Contact the system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0705 Unknown message *request*

Explanation: The displayed *request* is not supported by **sftp-server**.

System action: Command continues.

User response: Contact the system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0706 read error

Explanation: System call read() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0707 write error

Explanation: System call write() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0708 iqueue grows

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0709 msg_len *length* **< consumed** *bytes*

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0711 bad message from *client_address* **local user** *user_name*

Explanation: Internal error. A bad message was received from the client at *client_address* for local user *user_name*.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0712 read: *error_message*

Explanation: The read() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0713 write: *error_message*

Explanation: The write() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0714 Invalid log level "*log_level***"**

Explanation: The specified **sftp–server** log level value is not valid.

System action: The program continues.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid **sftp–server** log level values, and try the request again.

FOTS0715 Invalid log facility "*log_facility***"**

Explanation: The specified **sftp–server** log facility value is not valid.

System action: The program continues.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid **sftp–server** log facility values, and try the request again.

FOTS0716 Malformed SSH_CONNECTION variable: "*value*"

Explanation: The SSH_CONNECTION environment variable's value is malformed.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0717 select: *error_message*

Explanation: The select() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS0718 No user found for uid *UID*

Explanation: The getpwuid() system call was unable to get information about a user with UID *UID*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0801 pipe: *system error*

Explanation: System call pipe() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0802 socketpair: *system error*

Explanation: System call socketpair() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0803 fork: *system error*

Explanation: System call fork() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0804 dup2: *system error*

Explanation: System call dup2() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0805 exec: *path***:** *system error*

Explanation: System call exec() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0806 *error* **(***pathname***).**

Explanation: Error occurred when specifying *pathname* after '-b'.

System action: Command ends.

User response: Check to make sure that you use a valid path name.

FOTS0807 Filename already specified.

Explanation: You specified option '-b' more than once.

System action: Command ends.

User response: Check and make sure that you specify option '-b' only once.

FOTS0808 Invalid buffer size "*size***"**

Explanation: Buffer size can only be an integer between 1 and 2147483647(LONG_MAX).

System action: Command ends.

User response: Specify a valid buffer size and retry.

FOTS0809 Invalid number of requests "*number*"

Explanation: Number of requests can only be an integer between 1 and 2147483647(LONG_MAX).

System action: Command ends.

User response: Specify a valid number of requests and retry.

FOTS0810 Missing username

Explanation: User name is missing from the command line.

System action: Command ends.

User response: Check and make sure you issue a valid username on the command line.

FOTS0811 Missing hostname

Explanation: Host name is missing from the command line.

System action: Command ends.

User response: Check and make sure you issue a valid hostname on the command line.

FOTS0812 Couldn't wait for ssh process: *system error*

Explanation: System call waitpid() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0813 Shell exited abnormally

Explanation: The child process ended abnormally.

System action: Command continues.

User response: Contact the system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0814 Shell exited with status *status*

Explanation: The child process ended normally with the status listed above.

System action: Command continues.

User response: Contact the system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0815 Invalid path

Explanation: Internal error.

System action: Command continues.

User response: Contact your system programmer.

FOTS0816 ls: Invalid flag –*flag*

Explanation: You specified an invalid flag *flag* after the subcommand **ls**.

System action: Command continues.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a valid flag.

FOTS0817 Unterminated quote

Explanation: You specified quoted filename and the quotes are not closed.

System action: Command continues.

User response: Check and make sure the quotes are closed.

FOTS0818 Empty quotes

Explanation: You specified quoted filename and the file name is missing between the quotes.

System action: Command continues.

User response: Check and make sure to specify filename between the quotes.

FOTS0819 File "*filename*" not found.

Explanation: You specified a file that was not found.

System action: Command continues.

User response: Make sure the file exists before reissuing command.

FOTS0820 Multiple files match, but "*path*" is not a directory

Explanation: You attempted to upload more than one file but the target indicated by *path* was not a directory.

System action: Command continues.

User response: When uploading more than one file, ensure that the target *path* is a directory.

FOTS0821 Can't ls: "*path*" not found

Explanation: Internal error.

System action: Command continues.

User response: Contact the system programmer.

System programmer response: Follow local

procedures for reporting problems to IBM.

FOTS0822 Invalid command.

Explanation: You entered an invalid subcommand.

System action: Command continues.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a list of valid subcommands.

FOTS0823 You must specify at least one path after a *get or put* command.

Explanation: You omitted pathname after *get or put* command.

System action: Command continues.

User response: Check to make sure you specify at least one pathname after *get or put*.

FOTS0824 You must specify two paths after a *command* command.

Explanation: You specified only one pathname after the subcommand.

System action: Command continues.

User response: Check to make sure you specify two pathnames.

FOTS0825 You must specify a path after a *command* command.

Explanation: You omitted the pathname after the subcommand.

System action: Command continues.

User response: Check to make sure you did not omit the pathname.

FOTS0826 You must supply a numeric argument to the *cmd_string* command.

Explanation: You specified a non-numeric argument.

System action: Command continues.

User response: Check to make sure you specify a numeric argument.

FOTS0827 Can't change directory: Can't check target

Explanation: You can not change directory because the sftp-server protocol does not support remote file permission bits transferring.

System action: Command continues.

User response: Contact the system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0828 Can't change directory: "*dir*" is not a directory

Explanation: You can not change the directory because the argument specified after the subcommand **cd** is not a directory.

System action: Command continues.

User response: Check to make sure the argument you supply is a valid directory.

FOTS0829 Couldn't change local directory to "*dir*": *error*

Explanation: You can not change local directory because of the system error.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

FOTS0830 Couldn't create local directory "*dir*": *error*

Explanation: You can not create a local directory because of the system error.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

FOTS0831 Can't get current ownership of remote file "*pathname*"

Explanation: You can not get the ownership of the remote file because the sftp-server protocol does not support file ownership transferring.

System action: Command continues.

User response: Contact the system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0832 Couldn't get local cwd: *system error*

Explanation: You can not get local working directory because call to getcwd() failed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

FOTS0833 Couldn't fork: *system error*

Explanation: System call fork() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0834 Couldn't wait for child: *system error*

Explanation: System call waitpid() failed.

System action: Command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0835 Command not implemented

Explanation: The subcommand you specified is not implemented in the program.

System action: Command ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0836 *command number* **is not implemented**

Explanation: The specified interactive command is not implemented in the program.

System action: Command ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0837 Couldn't initialize connection to server

Explanation: Internal error.

System action: Command ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0838 Need cwd

Explanation: The program could not get the current working directory from the server.

System action: Command ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0839 Couldn't execute "*shell program***":** *system error*

Explanation: You specified interactive command '!' to invoke the local shell and the program failed to execute the local shell.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0840 Couldn't send packet: *system error*

Explanation: A call to write() failed while **sftp** was attempting to send packet to the server.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0841 Connection closed

Explanation: A call to read() failed while **sftp** was attempting to get packet from the server. Therefore, the connection between the client and the server was closed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0842 Couldn't read packet: *system error*

Explanation: A call to read() failed while **sftp** was attempting to get packet from the server.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0843 Received message too long *length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0844 ID mismatch (*received msg_id* **!=** *expected msg_id***)**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0845 Expected SSH2_FXP_STATUS(*packet type1***) packet, got** *packet type2*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0846 Expected SSH2_FXP_HANDLE(*handle1***) packet, got** *handle2*

Explanation: Internal error

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0847 Couldn't stat remote file: *error message*

Explanation: **sftp** failed to get the remote file information due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0848 Expected SSH2_FXP_ATTRS(*packet type1***) packet, got** *packet type2*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0849 Invalid packet back from SSH2_FXP_INIT (type *packet type***)**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0850 Couldn't close file: *error message*

Explanation: **sftp** failed to close the connection between the client and the server due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0851 Couldn't read directory: *error message*

Explanation: **sftp** failed to read the remote directory due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0852 Bad escaped character '*character***'**

Explanation: An invalid escaped character *character* was encountered after '\' in the file name.

System action: The program continues.

User response: Correct the file name and reissue the command.

FOTS0853 Couldn't delete file: *error message*

Explanation: **sftp** failed to delete the remote file due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0854 Couldn't create directory: *error message*

Explanation: **sftp** failed to create the remote directory due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0855 Couldn't remove directory: *error message*

Explanation: **sftp** failed to remove the remote directory due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0856 Couldn't setstat on "*path***":** *error message*

Explanation: **sftp** failed to set remote file attributes due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0857 Couldn't fsetstat: *error message*

Explanation: **sftp** failed to set remote file attributes due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0858 Couldn't canonicalise: *error_msg*

Explanation: Internal error.

System action: The program continues.

User response: Not applicable

System programmer response: Not applicable

FOTS0859 Expected SSH2_FXP_NAME(*packet type1***) packet, got** *packet type2*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0860 Got multiple names (*count***) from SSH_FXP_REALPATH**

Explanation: **sftp** received more than one remote real path.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0861 Couldn't rename file "*old_path***" to "***new_path***":** *error message*

Explanation: **sftp** failed to rename remote file due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0862 This server does not support the symlink operation

Explanation: The sftp server you connected to does not support the **ln** and **symlink** subcommands.

System action: The program continues.

User response: Do not use the **symlink** or **ln** subcommands.

FOTS0863 Couldn't readlink: *error message*

Explanation: **sftp** failed to read the remote symlink.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0864 Got multiple names (*count***) from SSH_FXP_READLINK**

Explanation: **sftp** received more than one symbolic names resolved for remote symlink.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0865 Cannot download a directory: *remote path*

Explanation: You can not download a remote directory.

System action: The program continues.

User response: Check to make sure that you do not specify a remote directory.

FOTS0866 **Couldn't open local file "***local path***" for writing:** *system error*

Explanation: Opening local file failed due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0867 **Unexpected reply** *message id*

Explanation: Received unexpected reply from the server while attempting to download remote file.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0868 **Received more data than asked for** *length of transferred data* **>** *buffer size*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0869 **Expected SSH2_FXP_DATA(***packet type1***) packet, got** *packet type2*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0870 **Transfer complete, but requests still in queue**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0871 **Couldn't read from remote file "***remote path***" :** *error message*

Explanation: sftp server failed to read from the remote file during downloading due to the displayed error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0872 **Couldn't write to "***local file***":** *system error*

Explanation: **sftp** failed to write to the local file during downloading due to the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0873 **Couldn't set mode on "***local file***":** *system error*

Explanation: **sftp** failed to change the mode of the local file due to the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0874 **Can't set times on "***local file***":** *system error*

Explanation: **sftp** failed to set the access and modification times of the local file due to the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0875 **Couldn't open local file "***local file***" for reading:** *system error*

Explanation: **sftp** failed to open the local file for reading (while attempting to upload the local file) due to the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time*

Library Reference for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0876 Couldn't fstat local file "*local file***": *system error***

Explanation: **sftp** failed to retrieve status information about the local file (while attempting to upload the local file) due to the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0877 Couldn't read from "*local file***": *system error***

Explanation: **sftp** failed to read from the local file (while attempting to upload the local file) due to the displayed system error.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0878 Unexpected ACK *message id*

Explanation: Internal error. Unexpected acknowledgment was received.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0879 Expected SSH2_FXP_STATUS(*packet type1***) packet, got** *packet type2*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0880 Can't find request for ID *request id*

Explanation: **sftp** failed to find the request from the request queue.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0881 Couldn't write to remote file "*filename***": *error_message***

Explanation: **sftp** failed to write to the remote file *filename* (while attempting to upload file) due to the displayed error message.

System action: The program continues.

User response: Correct the error, if possible, and attempt to upload the file again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0882 Couldn't close local file "*local file***": *system error***

Explanation: **sftp** failed to close the local file (after uploading the local file to the remote host) due to the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0883 Couldn't get handle: *error message*

Explanation: **sftp** failed to get handle sent from the server due to the displayed error message.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0884 skipping non-regular file *file_name*

Explanation: While processing file to be uploaded, a non-regular file *file_name* was encountered and was ignored by **sftp**.

System action: The program continues.

User response: Check to make sure not to upload a non-regular file.

FOTS0885 **stat** *path*: *system_error*

Explanation: System call stat() failed on *path* due to the displayed system error.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0886 **Batch file already specified.**

Explanation: You specified option '-b' more than once.

System action: Command ends.

User response: Check and make sure that you specify option '-b' only once.

FOTS0887 **Couldn't symlink file "***old_path***" to "***new_path***":** *error message*

Explanation: **sftp** failed to symlink from *old_path* to *new_path* due to the displayed error.

System action: The program continues.

User response: If unable to resolve based on the displayed error, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0888 **Cannot download non-regular file:** *file_name*

Explanation: You were trying to download a non-regular file *file_name* from the remote host. This cannot be performed by **sftp**.

System action: The program continues.

User response: Check and make sure not to download a non-regular file.

FOTS0889 *file_name* **is not a regular file**

Explanation: You were trying to download a non-regular file *file_name* from the remote host. This cannot be performed by **sftp**.

System action: The program continues.

User response: Check and make sure not to download a non-regular file.

FOTS0890 **Outbound message too long** *msg_len*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0891 **Read packet:** *system_error*

Explanation: System call read() failed due to the displayed system error.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0893 **remote_glob failed with return code** *return code***.**

Explanation: A call to the OpenSSH function remote_glob failed. The function's return value is displayed with this message.

System action: If running in an interactive session, the command continues. If running in batchmode, the command ends.

User response: Internal error. Contact your system administrator to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0894 *command*: **Invalid flag –***flag*

Explanation: You specified an invalid flag *flag* after the subcommand *command*.

System action: Command continues.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a valid flag.

FOTS0895 **string too long**

Explanation: **sftp** encountered a command string that was too long.

System action: The program continues.

User response: Shorten the command string length and try the request again. If unable to resolve, contact your system programer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0896 **Unterminated quoted argument**

Explanation: **sftp** encountered an unterminated quoted argument while parsing a command string.

System action: The program continues.

User response: Verify quoted arguments are properly terminated and try the request again. If unable to

resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0897 Unknown ls sort type

Explanation: You specified an unknown **ls** sort type.

System action: The program ends.

User response: Check *IBM Ported Tools for z/OS User's Guide* for a valid **ls** sort type.

FOTS0901 Couldn't obtain random bytes (error *error*)

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0902 fstat for key file *file_name* failed: *system_error*

Explanation: System call fstat() failed on key file *file_name* due to the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0903 key_load_private_rsa1: RSA_blinding_on failed

Explanation: A call to OpenSSL function RSA_blinding_on() failed.

System action: The program continues.

User response: Check OpenSSL function RSA_blinding_on() for more information.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0904 key_load_private_pem: RSA_blinding_on failed

Explanation: A call to OpenSSL function RSA_blinding_on() failed.

System action: The program continues.

User response: Check OpenSSL function RSA_blinding_on() for more information.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0905 buffer_put_bignum2_ret: negative numbers not supported

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0906 buffer_put_bignum2_ret: BN too small

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0907 ssh1_3des_cbc: no context

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0908 ssh_rijndael_iv: no context

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0909 ssh_aes_ctr_iv: no context

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0910 Authentication response too long: *length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0914 **mkstemp("***temp file***"):** *system error*

Explanation: Failed to open/create temp file due to the displayed system error.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0915 *function***: UsePrivilegeSeparation=yes and Compression=yes not supported**

Explanation: ssh does not support when you specify both UsePrivilegeSeparation=yes and Compression=yes at the same time.

System action: The program continues.

User response: Check to make sure that you do not specify UsePrivilegeSeparation=yes and Compression=yes at the same time.

FOTS0916 **Error writing to authentication socket.**

Explanation: Failure occurred while writing to authentication socket.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0917 **Error reading response length from authentication socket.**

Explanation: Failure occurred while reading from authentication socket.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0918 **Error reading response from authentication socket.**

Explanation: Failure occurred while reading from authentication socket.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0919 **Authentication response too long:** *length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0920 **Bad authentication reply message type:** *type*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0921 **Too many identities in authentication reply:** *number*

Explanation: Received too many identities in reply.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0922 **Bad authentication response:** *response type*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0924 **Bad response from authentication agent:** *response type*

Explanation: Received unsupported response from ssh-agent.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0925 **open** *filename* **failed:** *system error***.**

Explanation: Failure occurred while attempting to open the key file. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time*

Library Reference for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0926 **write to key file** *filename* **failed:** *system error*

Explanation: Failure occurred while attempting to write into a key file. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0927 **passphrase too short: have** *number* **bytes, need > 4**

Explanation: The new passphrase is too short. **ssh–keygen** does not allow passphrases that are less than or equal to 4 bytes.

System action: The program ends.

User response: Check to make sure that you enter a passphrase greater than 4 bytes long. Refer to *IBM Ported Tools for z/OS User's Guide* for an explanation of a valid passphrase.

System programmer response: Not applicable.

FOTS0928 **key file** *filename* **too large**

Explanation: The RSA key file *filename* is too large.

System action: The program continues.

User response: Verify that the file *filename* is a valid RSA key file, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0929 **fdopen** *filename* **failed:** *system error***.**

Explanation: Failure occurred while attempting to open the file for write. The system error is displayed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0930 **key_save_private: cannot save key type** *type*

Explanation: The displayed key type can not be saved.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Not applicable

FOTS0931 **fdopen failed:** *system error*

Explanation: Failure occurred while attempting to open the file for read. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0932 **PEM_read_PrivateKey: mismatch or unknown EVP_PKEY save_type** *save_type*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0934 **@ WARNING: UNPROTECTED PRIVATE KEY FILE! @ Permissions 0***permission bits* **for '***file name***' are too open. It is recommended that your private key files are NOT accessible by others. This private key will be ignored.**

Explanation: The permission bits of your key file is too open and that makes your key file insecure.

System action: The program continues.

User response: Check to make sure that your private key file is only readable by you.

FOTS0939 **bad permissions: ignore key:** *file name*

Explanation: The key file is readable by others.

System action: The program continues.

User response: Check to make sure that the private key file is only readable by you.

FOTS0941 save_private_key_rsa: bad cipher

Explanation: The cipher used to encrypt private keys is not supported.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0942 buffer_put_bignum_ret: BN_bn2bin() failed: oi *length* **!= bin_size** *size*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0943 buffer_get_bignum_ret: cannot handle BN of size *bytes*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0944 buffer_get_bignum_ret: input buffer too small

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0945 buffer_put_bignum2_ret: BN_bn2bin() failed: oi *length* **!= bin_size** *size*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0946 buffer_get_bignum2_ret: cannot handle BN of size *bytes*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0947 buffer_get_string_ret: bad string length *number*

Explanation: Internal error. Received string too long.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0948 buffer_put_cstring: s == NULL

Explanation: s is the input string to function buffer_put_cstring(). s cannot be an empty string.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0949 buffer_append_space: len *length* **not supported**

Explanation: Appended space cannot be greater than 1048576 bytes.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0950 buffer_append_space: alloc *number* **not supported**

Explanation: Cannot allocate buffer of size greater than 10485760 bytes.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0951 buffer_get_ret: trying to get more bytes *length* **than in buffer** *size available*

Explanation: The size of the available buffer is not big enough for the string.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0952 buffer_consume: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0953 buffer_consume_end: trying to get more bytes than in buffer

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0954 buffer_get_string_bin_ret: bad string length *string_length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0955 buffer_get_short: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0956 get_socket_ipaddr: getnameinfo *flag* **failed**

Explanation: A call to getnameinfo() failed. *flag* is the argument of getnameinfo().

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0957 getsockname failed: *system error*

Explanation: A call to getsockname() failed with the displayed system error.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0958 get_remote_hostname: getnameinfo NI_NUMERICHOST failed

Explanation: A call to getnameinfo() failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of argument NI_NUMERICHOST. Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0959 get_sock_port: getnameinfo NI_NUMERICSERV failed

Explanation: A call to getnameinfo() failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of argument NI_NUMERICSERV. Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0960 channel *channel identifier***: wfd** *write_fd* **is not a tty?**

Explanation: The write file descriptor of the channel is not associated with a terminal.

System action: The program continues.

User response: Check your command line options to see whether you need a tty. If the code sets were changed for the terminal, for example by issuing the **chcp** command, conversion may not be performed properly. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0961 X11 fake_data_len *length* **!= saved_data_len** *length*

Explanation: During X11 forwarding, fake data length is not equal to the saved data length.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0962 **accept:** *system error*

Explanation: A call to accept() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0963 **setsockopt SO_REUSEADDR fd** *file_descriptor***:** *system error*

Explanation: A call to setsockopt() failed. The system error is displayed. SO_REUSEADDR is one of the arguments of setsockopt().

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0964 **accept from auth socket:** *system error*

Explanation: A call to accept() failed. Authentication agent socket failed to accept the connection from the client. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0965 **getsockopt SO_ERROR failed**

Explanation: A call to getsockopt() failed. *SO_ERROR* is one of the arguments of getsockopt().

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0966 **No forward host name.**

Explanation: Port forwarding host name is NULL.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0967 **Forward host name too long.**

Explanation: The size of the forwarding host name is greater than 255.

System action: The program continues.

User response: Check to make sure that you do not specify a host name greater than 255. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0968 **channel_setup_fwd_listener: getnameinfo failed**

Explanation: A call to getnameinfo() failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0969 **setsockopt SO_REUSEADDR:** *system error*

Explanation: A call to setsockopt() failed. The system error is displayed. *SO_REUSEADDR* is one of the arguments of setsockopt().

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0970 **bind:** *system error*

Explanation: A call to bind() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0971 **listen:** *system error*

Explanation: A call to listen() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local

procedures for reporting problems to IBM.

FOTS0972 **channel_setup_fwd_listener: cannot listen to port:** *port*

Explanation: Port forwarding failed to listen to the displayed port.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0973 **connect_to** *hostname***: unknown host (***system error***)**

Explanation: A call to getaddrinfo() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0974 **connect_to: getnameinfo failed**

Explanation: A call to getnameinfo() failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0975 **socket:** *system error*

Explanation: A call to socket() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0976 **connect_to** *host name* **port** *service name***:** *system error*

Explanation: A call to connect() failed and the system error is displayed. *host name* and *service name* are the host name and the service location of the socket to which a connection was attempting. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0977 **connect_to** *host* **port** *port***: failed.**

Explanation: Failed to connect to *host* on *port*.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0978 **WARNING: Server requests forwarding for unknown listen_port** *listen_port*

Explanation: Internal error occurred. The displayed *listen_port* is not permitted for forwarding.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0979 **getaddrinfo:** *system error*

Explanation: A call to getaddrinfo() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

| **FOTS0980** **channel_setup_fwd_listener: getaddrinfo(***address***):** *error_message*

| **Explanation:** The getaddrinfo() system call failed. The system error is displayed with the message.

| **System action:** The program continues.

| **User response:** Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

| **System programmer response:** Take appropriate action based on the system error.

| **FOTS0981** **setsockopt IPV6_V6ONLY:** *system error*

| **Explanation:** A call to setsockopt() failed. IPV6_V6ONLY is one of the arguments of setsockopt(). The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0982 Failed to allocate internet-domain X11 display socket.

Explanation: The number of internet-domain X11 display sockets is greater than 1000.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0983 x11_request_forwarding_with_spoofing : different $DISPLAY already forwarded

Explanation: Unable to complete the X11 forwarding request because a different display has already been forwarded.

System action: The program continues.

User response: Verify that the value of your DISPLAY environment variable is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0984 socket: *system error*

Explanation: A call to socket() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0985 connect *path_name***:** *system error*

Explanation: A call to connect() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0986 DISPLAY not set.

Explanation: Environment variable DISPLAY is not set.

System action: The program continues.

User response: Refer to ssh in *IBM Ported Tools for z/OS User's Guide* on how to set environment variable DISPLAY. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0987 Could not parse display number from DISPLAY: *display*

Explanation: A call to sscanf() failed. UNIX domain display number cannot be parsed from environment variable DISPLAY *display*.

System action: The program continues.

User response: Refer to **ssh** in *IBM Ported Tools for z/OS User's Guide* on how to set environment variable DISPLAY. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0988 Could not find ':' in DISPLAY: *display*

Explanation: Did not find ':' in environment variable DISPLAY *display*.

System action: The program continues.

User response: Refer to ssh in *IBM Ported Tools for z/OS User's Guide* on how to set environment variable DISPLAY. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0989 *function***: unexpected data on ctl fd**

Explanation: Unexpected data read from the control file descriptor. The error occurred in *function*.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0990 *host_name***: unknown host. (***system error***)**

Explanation: A call to getaddrinfo() failed. The *host_name* is unknown. The system error is displayed.

System action: The program continues.

User response: Check to make sure the host name specified by the DISPLAY environment variable is valid. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0991 connect *host_name* **port** *port***:** *system error*

Explanation: A call to connect() failed. Failure occurred while attempting to connect to *host_name* on *port*. The system error is displayed.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0992 Warning: ssh server tried agent forwarding.

Explanation: The ssh configuration option ForwardAgent was disabled but ssh server requested a connection to the authentication agent.

System action: The program continues.

User response: Enable ForwardAgent option in ssh_config or on the command line.

FOTS0993 Warning: ssh server tried X11 forwarding.

Explanation: The ssh configuration option ForwardX11 was disabled but ssh server requested an X11 channel.

System action: The program continues.

User response: Enable ForwardX11 option in ssh_config or on the command line.

FOTS0994 deny_input_open: type *request type*

Explanation: Internal error. The *request type* is unsupported.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0995 Warning: this is probably a break–in attempt by a malicious server.

Explanation: Internal error or you requested to open an X11/Agent forwarding channel without enabling ForwardX11/ForwardAgent.

System action: The program continues.

User response: Enable ForwardX11 or ForwardAgent option in ssh_config or on the command line. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0996 channel_new: internal error: channels_alloc *number of allocations* **too big.**

Explanation: Internal error occurred. The number of allocated channels is greater than 10000.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0997 cannot happen: SSH_CHANNEL_LARVAL

Explanation: Channel type SSH_CHANNEL_LARVAL cannot happen with SSH Protocol 2.0

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0998 cannot happen: OUT_DRAIN

Explanation: Channel type OUT_DRAIN cannot happen with SSH Protocol 1.3

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS0999 channel_still_open: bad channel type *channel_type*

Explanation: Channel is still open with invalid channel type.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1001 evp_crypt: EVP_Cipher failed during discard

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local

| procedures for reporting problems to IBM.

FOTS1002 **channel_find_open: bad channel type** *channel_type*

Explanation: Found a channel open with invalid channel type.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1003 **channel_open_message: bad channel type** *channel_type*

Explanation: Channel with invalid channel type is open.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1004 **channel_activate for non–larval channel** *channel_id***.**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1005 **channel** *channel_id***: decode socks4: len** *expected length* **> have** *actual length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1006 **cannot happen: istate == INPUT_WAIT_DRAIN for proto 1.3**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1007 **channel_add_permitted_opens: too many forwards**

Explanation: A request for forwarding an application over a new channel was denied because the internal maximum of forwarded channels has been reached.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the authorized keys file permitopen option. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1009 **connect_to: F_SETFL:** *system error*

Explanation: A call to fcntl() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1010 **x11_request_forwarding: bad authentication data:** *data*

Explanation: Internal error or your xauth program generated invalid authentication data.

System action: The program ends.

User response: Check xauth program to make sure it generates valid authentication data or contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

| **FOTS1011** **Warning: use of DES is strongly discouraged due to cryptographic weaknesses**

Explanation: You are using cipher type DES and it is strongly discouraged due to cryptographic weaknesses.

System action: The program continues.

| **User response:** Refer to **ssh** in *IBM Ported Tools for*
| *z/OS User's Guide* for an explanation of DES.

FOTS1012 **cipher_cleanup: EVP_CIPHER_CTX_cleanup failed**

Explanation: A call to OpenSSL function EVP_CIPHER_CTX_cleanup() failed.

System action: The program continues.

User response: Check OpenSSL function

EVP_CIPHER_CTX_cleanup() for more information.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1013 ssh1_3des_cbc: no context

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1014 ssh_rijndael_cbc: no context

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1015 cipher_init: key length *length* **is insufficient for** *cipher type***.**

Explanation: Internal error occurred. The length of the key is insufficient for the displayed *cipher type*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1016 cipher_init: iv length *length* **is insufficient for** *cipher type*

Explanation: Internal error occurred. IV length is not sufficient for the displayed *cipher type*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1017 cipher_init: EVP_CipherInit failed for *cipher type*

Explanation: A call to OpenSSL function EVP_CipherInit() failed.

System action: The program ends.

User response: Check OpenSSL function EVP_CipherInit() for more information. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1018 cipher_init: set keylen failed (*key_length* **->** *key_length setting to***)**

Explanation: A call to OpenSSL function EVP_CIPHER_CTX_set_key_length() failed.

System action: The program ends.

User response: Check OpenSSL function EVP_CIPHER_CTX_set_key_length() for more information. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1019 cipher_init: EVP_CipherInit: set key failed for *cipher type*

Explanation: A call to OpenSSL function EVP_CipherInit() failed.

System action: The program ends.

User response: Check OpenSSL function EVP_CipherInit() for more information. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1020 cipher_encrypt: bad plaintext length *length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1021 evp_crypt: EVP_Cipher failed

Explanation: A call to OpenSSL function EVP_Cipher() failed.

System action: The program ends.

User response: Check OpenSSL function EVP_Cipher() for more information. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1022 ssh_rijndael_cbc: bad len *length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1023 *function*: **wrong iv length** *expected length* **!=** *actual length*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1024 *function*: **no rijndael context**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1025 *function*: **bad 3des iv length:** *length*

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1026 *function*: **no 3des context**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1027 *function*: **bad cipher** *cipher_type*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1028 **mac_compute: unknown MAC type**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1029 **mac** *MAC_name* **len** *MAC_length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1030 **mac_compute: mac too long** *MAC_length* *maximum_MAC_length*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1031 **No available ciphers found.**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1032 **Bad compression level** *number*.

Explanation: You specified an invalid compression level.

System action: The program ends.

User response: Check your ssh_config file or command line to make sure you specify a valid CompressionLevel.

FOTS1033 **buffer_compress: deflate returned** *status*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1034 **buffer_uncompress: inflate returned** *status*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1035 **detect_attack: bad length** *number*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1036 **Bad prime description in line** *line_number*

Explanation: File moduli or primes contains invalid prime description in *line_number*.

System action: The program continues.

User response: Check moduli or primes to make sure prime descriptions are valid.

FOTS1037 **parse_prime: BN_new failed**

Explanation: A call to OpenSSL function BN_new() failed. BN_new() allocates and initializes a BIGNUM structure.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1038 *function***: BN_new failed**

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1039 **WARNING: line** *line_num* **disappeared in** *file***, giving up**

Explanation: Internal error or the displayed *line_num* is missing from file primes.

System action: The program continues.

User response: Check your primes file to make sure the displayed *line_num* exists. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1040 **dh_gen_key: dh->p == NULL**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1041 **dh_gen_key: group too small:** *bits* **(2*need** *bits***)**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1042 **dh_gen_key: BN_new failed**

Explanation: A call to OpenSSL function BN_new() failed. BN_new() allocates and initializes a BIGNUM structure.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1043 **dh_gen_key: BN_rand failed**

Explanation: A call to OpenSSL function BN_rand() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1044 **DH_generate_key**

Explanation: A call to OpenSSL function DH_generate_key() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1045 **dh_gen_key: too many bad keys: giving up**

Explanation: Internal error. Too many invalid public keys are generated.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1046 dh_new_group_asc: DH_new

Explanation: A call to OpenSSL function DH_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1047 BN_hex2bn p

Explanation: A call to OpenSSL function BN_hex2bn() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1048 BN_hex2bn g

Explanation: A call to OpenSSL function BN_hex2bn() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1049 dh_new_group: DH_new

Explanation: A call to OpenSSL function DH_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1050 protocol error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1051 mac_init: no key

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1052 mac_compute: mac too long

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1053 ssh_msg_send: write

Explanation: Internal error. Partial data was written from the buffer into the file descriptor.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1054 add_host_to_hostfile: host_hash failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1055 ssh_msg_recv: read: header *bytes*

Explanation: Internal error. Partial data was read from the file descriptor into the buffer.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1056 ssh_msg_recv: read: bad msg_len *bytes*

Explanation: Internal error. The data received was too long.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1057 ssh_msg_recv: read: *bytes* **!= msg_len**

Explanation: Internal error. Partial data was read from the file descriptor into the buffer.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1058 add_host_to_hostfile: saving key in *file* **failed**

Explanation: Adding keys to host file failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1059 no key to look up

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1060 write_bignum: BN_bn2dec() failed

Explanation: A call to OpenSSL function BN_bn2dec() failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1061 key_read: uudecode *key* **failed**

Explanation: Internal error. A call to uudecode() failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1062 key_read: key_from_blob *key* **failed**

Explanation: Internal error. A call to key_from_blob() failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1063 key_read: type mismatch: encoding error

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1064 key_write: failed for RSA key

Explanation: Internal error. A call to OpenSSL function BN_bn2dec() failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1065 key_from_blob: cannot handle type *key_type*

Explanation: Internal error. The displayed key type is not valid.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1066 key_from_blob: remaining bytes in key blob *bytes*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1067 key_to_blob: key == NULL

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1068 key_to_blob: unsupported key type *type*

Explanation: The displayed key *type* is not valid.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1069 key_sign: illegal key type *type*

Explanation: Internal error. The displayed key *type* is not valid.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1070 key_verify: illegal key type *type*

Explanation: The displayed key *type* is not valid.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1071 key_new: RSA_new failed

Explanation: A call to OpenSSL function RSA_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1072 key_new: BN_new failed

Explanation: A call to OpenSSL function BN_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1073 host_hash: __b64_ntop failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1074 key_new: DSA_new failed

Explanation: A call to OpenSSL function DSA_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1079 key_new: bad key type *type*

Explanation: Internal error. The displayed key *type* is not valid.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1080 key_new_private: BN_new failed

Explanation: A call to OpenSSL function BN_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1085 key_from_private: BN_copy failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1086 key_free: key is NULL

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1087 key_free: bad key type *type*

Explanation: Internal error. The displayed key *type* is not valid.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1088 key_equal: bad key type *type*

Explanation: Internal error. The displayed key *type* is not valid.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1089 **key_fingerprint_raw: bad digest type** *MAC_algorithm*

Explanation: The displayed *MAC_algorithm* is not supported.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1090 **key_fingerprint_raw: bad key type** *type*

Explanation: Internal error. The displayed key *type* is not valid.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1091 **key_fingerprint_raw: blob is null**

Explanation: internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1092 **key_fingerprint: null from key_fingerprint_raw()**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1093 **key_fingerprint_ex: bad digest representation** *fingerprint*

Explanation: Internal error. The displayed *fingerprint* is not valid.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1094 **key_read: bad key type:** *type*

Explanation: The key type *type* is not valid.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1095 **rsa_generate_private_key: key generation failed.**

Explanation: A call to OpenSSL function RSA_generate_key() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1096 **dsa_generate_private_key: DSA_generate_parameters failed**

Explanation: A call to OpenSSL function DSA_generate_parameters() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1097 **dsa_generate_private_key: DSA_generate_key failed.**

Explanation: A call to OpenSSL function DSA_generate_key() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1098 **dsa_generate_private_key: NULL.**

Explanation: A call to OpenSSL function DSA_generate_key() generated a NULL private DSA key.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1099 **key_generate: unknown type** *key_type*

Explanation: You specified an invalid key type on the command line.

System action: The program continues.

User response: Check to make sure you specify a valid key type on the command line.

FOTS1101 **key_from_private: unknown type** *key_type*

Explanation: The *key_type* is not valid. The error is usually caused by an invalid key type specified after option –t. This message can also be displayed for an internal error.

System action: The program ends.

User response: Check to make sure you specify a valid key type after option –t. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1102 **key_demote: RSA_new failed**

Explanation: A call to OpenSSL function RSA_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1103 **key_demote: BN_dup failed**

Explanation: A call to OpenSSL function BN_dup() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1105 **key_demote: DSA_new failed**

Explanation: A call to OpenSSL function DSA_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1108 *function*: **bad server modulus (len** *length*)

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1109 *function*: **bad host modulus (len** *length*)

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1110 **bad kex md size** *MD_size*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1111 **Hm, kex protocol error: type** *protocol_type* **seq** *packet_id*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1112 **kex_send_kexinit: no kex, cannot rekey**

Explanation: The kex structure is NULL.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1113 **kex_send_kexinit: kex proposal too short**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1114 **kex_input_kexinit: no kex, cannot rekey**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1115 Unsupported key exchange *type*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1116 no matching cipher found: client *proposal* **server** *proposal*

Explanation: Did not find the cipher that the client and the server both support.

System action: The program ends.

User response: Reissue the command with specifying the cipher that the server supports.

FOTS1117 matching cipher is not supported: *cipher*

Explanation: The *cipher* is not supported by the daemon.

System action: The program ends.

User response: Reissue the command with specifying the cipher that the server supports either in ssh_config file or on the command line.

FOTS1118 no matching mac found: client *proposal* **server** *proposal*

Explanation: Did not find the MAC that the client and the server both support.

System action: The program ends.

User response: Reissue the command with specifying the MAC that the server supports either in ssh_config file or on the command line.

FOTS1119 unsupported mac *MAC*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1120 no matching comp found: client *proposal1* **server** *proposal2*

Explanation: Did not find the Compression option that the client and the server both support.

System action: The program ends.

User response: Reissue the command specifying the Compression option that the server supports either in ssh_config file or on the command line.

FOTS1121 unsupported comp *compression*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1122 Unable to negotiate a key exchange method

Explanation: Did not find the key–exchange algorithm that the client and the server both support.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1123 bad kex alg *algorithm*

Explanation: The displayed key-exchange *algorithm* is not supported.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1124 no hostkey alg

Explanation: Did not find the key type that the client and the server both support.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1125 bad hostkey alg '*key_type***'**

Explanation: The displayed *key_type* is not supported.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1129 cannot decode server_host_key_blob

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1130 type mismatch for decoded server_host_key_blob

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1131 cannot verify server_host_key

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1132 server_host_key verification failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1133 dh_server_pub == NULL

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1134 kexdh_client: BN_new failed

Explanation: Internal error. A call to OpenSSL function BN_new() failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

System action: The program continues.

FOTS1135 key_verify failed for server_host_key

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1136 Cannot load hostkey

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1137 Unsupported hostkey type *key_type*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1138 dh_client_pub == NULL

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1139 kexdh_server: BN_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1144 BN_new

Explanation: The BN_new() function failed.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1146 DH_GEX group out of range: *min* **!<** *num_bits* **!<** *max*

Explanation: The big number returned by BN_new is malformed.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1147 cannot decode server_host_key_blob

Explanation: Unable to decode the server host key blob.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1148 type mismatch for decoded server_host_key_blob

Explanation: The key received from the server is not the proper type.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1149 cannot verify server_host_key

Explanation: Unable to verify the server host key.

System action: The program ends.

User response: Verify that the public key for the remote host is accurate. Contact the system programmer of the server for further assistance.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1150 server_host_key verification failed

Explanation: Server host key verification failed.

System action: The program ends.

User response: Verify that the public key for the remote host is accurate. Contact the system programmer of the server for further assistance.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1151 dh_server_pub == NULL

Explanation: The value of dh_server_pub generated by BN_new is NULL.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1152 kexgex_client: BN_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1153 key_verify failed for server_host_key

Explanation: The key_verify() function failed for the given server_host_key.

System action: The program ends.

User response: Verify that the public key for the remote host is accurate. Contact the system programmer of the server for further assistance.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1154 Cannot load hostkey

Explanation: Unable to load the host key.

System action: The program ends.

User response: Verify that the host key exists on your system or contact the system programmer for further assistance.

System programmer response: Verify host key file. If problem cannot be resolved follow local procedures for reporting problems to IBM.

FOTS1155 Unsupported hostkey type *keytype*

Explanation: The type of host key specified is not supported.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1156 protocol error during kex, no DH_GEX_REQUEST: *type*

Explanation: Packet received does not match recognized request types.

System action: The program ends.

User response: Verify connectivity and ssh server status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1157 **DH_GEX_REQUEST, bad parameters:** *min* **!<** *num_bits* **!<** *max*

Explanation: The number of bits received in a server packet is incorrect.

System action: The program ends.

User response: Verify connectivity and ssh server status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1158 **dh_client_pub == NULL**

Explanation: BN_new() function call returned NULL.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1159 **kexgex_server: BN_new failed**

Explanation: BN_new() function call failed.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1165 **fatal_remove_cleanup: no such cleanup function: 0x***proc* **0x***context*

Explanation: Cleanup error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1168 **Unrecognized internal syslog level code** *level*

Explanation: Invalid syslog level specified. An internal error has occurred.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1169 **Unrecognized internal syslog facility code** *facility*

Explanation: Invalid syslog facility specified. An internal error has occurred.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1171 **fcntl(***fd***, F_GETFL, 0):** *error_code*

Explanation: fcnt() system call failed.

System action: Command continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1173 **getsockopt TCP_NODELAY:** *error_code*

Explanation: getsockopt() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1174 **setsockopt TCP_NODELAY:** *error_code*

Explanation: setsockopt() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1175 **Warning: identity keysize mismatch: actual** *keysize1***, announced** *keysize2*

Explanation: The agent's RSA identity contains a keysize mismatch.

System action: The program continues.

User response: Verify that the agent's RSA identity is valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1176 Compatibility with ssh protocol version 1.0 no longer supported.

Explanation: RSA authentication challenge not supported with SSH protocol version 1.0.

System action: The program continues.

User response: Use a newer version of SSH protocol version 1, and try the request again.

FOTS1177 Agent admitted failure to authenticate using the key.

Explanation: The agent failed the RSA authentication challenge.

System action: The program continues.

User response: Verify that the agent's RSA identity is valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1178 Agent admitted failure to sign using the key.

Explanation: The agent failed to generate a signature using a key.

System action: The program continues.

User response: Verify that the agent's identities are valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1179 SSH_AGENT_FAILURE

Explanation: The agent indicated a failure to handle a request.

System action: The program continues.

User response: Verify that the agent's identities, connection, and request are valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1180 parse_tty_modes: unknown opcode *opcode*

Explanation: The tty mode opcode *opcode* is undefined.

System action: The program continues.

User response: Verify the tty mode opcode, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1181 parse_tty_modes: n_bytes_ptr != n_bytes: *bytes1 bytes2*

Explanation: The tty mode packet contained the incorrect number of bytes.

System action: The program continues.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1182 Value "*value***" not valid for environment variable** *environment_variable*

Explanation: The value *value* for environment variable *environment_variable* is not valid.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid environment variable values, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1183 Couldn't open /dev/null: *error_message*

Explanation: The open() system call failed to open /dev/null. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1184 dup2: *error_message*

Explanation: The dup2() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1185 *function*: **out of memory (allocating** *size* **bytes)**

Explanation: Unable to allocate requested number of bytes. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1186 **Finished discarding for** *ip_address*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1187 **Bad packet length** *packet_length*.

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1188 **padding error: need** *needed_size* **block** *block_size* **mod** *modulus*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1189 **Corrupted MAC on input.**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1190 **internal error need** *needed_size*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1237 **Could not create directory** *dirname*: *error_message*

Explanation: The directory *dirname* could not be created. A call to mkdir() failed. The system error is displayed with this message.

System action: The program continues.

User response: Make sure you have appropriate authority to create the directory. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1238 **Could not request local forwarding.**

Explanation: A local forwarding request has failed.

System action: The program continues.

User response: Check for additional error messages displayed with this message, and take appropriate action. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the error messages displayed with this message. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1239 **setrlimit failed:** *system error*

Explanation: A call to setrlimit() failed while attempting to set RLIMT_CORE to zero. The system error is displayed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1240 **Too many identity files specified (max** *max*)

Explanation: The maximum number of authentication identity files and key ring certificates (*max*) that can be specified in configuration files or the command line has been exceeded.

System action: The program ends.

User response: Reissue the command with a smaller number of identity files or key ring certificates.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1241 Too high debugging level.

Explanation: For **ssh**, the -v (verbose) option was specified too many times. For **sshd**, the -d (debug) option was specified too many times.

System action: The program ends.

User response: Reissue the command with less instances of -v (or -d) specified.

FOTS1242 Cannot fork into background without a command to execute.

Explanation: The **ssh** -f option was specified without a command to execute.

System action: The program ends.

User response: Reissue **ssh** with a command or without the -f option.

FOTS1243 Can't open user config file *filename***:** *system error*

Explanation: ssh was unable to open the user configuration file *filename*. The system error is displayed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1244 Compression level must be from 1 (fast) to 9 (slow, best).

Explanation: An invalid compression level was specified.

System action: The program ends.

User response: Reissue the command with an appropriate compression level.

FOTS1245 daemon() failed: *system error*

Explanation: Either a call to fork() or setsid() failed while ssh was attempting to continue running in the background. The system error is displayed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1246 Request for subsystem '*command***' failed on channel** *channel*

Explanation: The ssh daemon rejected the client's request for subsystem *command* on channel *channel*.

System action: The program ends.

User response: Verify sshd is configured to use the subsystem or contact your system programmer.

System programmer response: Verify sshd is configured to use the subsystem.

FOTS1247 dup() in/out/err failed: *system error*

Explanation: A call to dup() for stdin, stdout or stderr failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1248 No support for forwarding GSSAPI credentials.

Explanation: ssh on z/OS does not provide support for forwarding GSS-API credentials.

System action: The program continues.

User response: Issue the command without the options to enable or disable forwarding GSS-API credentials (-k or -K for ssh).

System programmer response: None.

FOTS1252 The SSH client cannot be run under OMVS.

Explanation: The SSH client cannot be run under OMVS (a 3270 session) due to password visibility issues.

System action: The program ends.

User response: Reissue the command from a non-OMVS environment, for example, a TCP/IP session.

System programmer response: Not applicable

FOTS1254 *function* **listen():** *error_message*

Explanation: The listen() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error.

If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1255 load_public_identity_files: getpwuid failed

Explanation: The getpwuid() system call failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1256 load_public_identity_files: gethostname: *error_message*

Explanation: The gethostname() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1257 env_permitted: name '*environment_variable...***' too long**

Explanation: The environment variable name *environment_variable...* is too long.

System action: The program ends.

User response: Verify that your environment variable names do not exceed 1023 bytes, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1258 Control socket connect(*control_path***):** *error_message*

Explanation: The connect() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1259 open(/dev/null): *error_message*

Explanation: The open() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1260 dup2: *error_message*

Explanation: The dup2() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1261 *function***: msg_send**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1262 *function***: msg_recv**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1263 *function***: wrong version**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1264 Connection to master denied

Explanation: The master process denied access to its shared connection.

System action: The program ends.

User response: Verify that the control path is valid and that the master process permits access to its shared connection, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config ControlPath and ControlMaster keywords. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1265 silly mux_command *command_value*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1266 *function***: send fds failed**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1267 *function***: read** *error_message*

Explanation: The read() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1268 *function***: master returned too much data (**_actual_data_length_ **>** _expected_data_length_**)**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1272 Control socket connect(*control_path***):** *error_message*

Explanation: The connect() system call failed. The system error is displayed with the message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1273 Warning: Identity file *filename* **not accessible:** *error_message***.**

Explanation: The **ssh** –i option is set to a file that is not accessible. The system error is displayed with the message.

System action: The program continues.

User response: Verify that the value for the **ssh** –i option is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** –i option. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1274 Bad local forwarding specification '*value***'**

Explanation: The **ssh** –L option is set to a bad value *value*.

System action: The program ends.

User response: Verify that the value for the **ssh** –L option is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** –L option. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1275 Bad remote forwarding specification '*value***'**

Explanation: The **ssh** –R option is set to a bad value *value*.

System action: The program ends.

User response: Verify that the value for the **ssh** –R option is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** –R option. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1276 Invalid multiplex command.

Explanation: The **ssh** –O option is set to an unsupported value.

System action: The program ends.

User response: Verify that the value for the **ssh** –O option is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** –O option. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1277 gethostname: *error_message*

Explanation: The gethostname() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1278 No ControlPath specified for "–O" command

Explanation: The **ssh** –O option was specified, but no control path was set via the **ssh** –S option or the ssh_config ControlPath keyword.

System action: The program ends.

User response: Verify that a control path is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** options and the ssh_config keywords.

FOTS1279 Could not request local forwarding.

Explanation: A local forwarding request has failed.

System action: The program ends.

User response: Check for additional error messages displayed with this message, and take appropriate action. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the error messages displayed with this message. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1280 Could not request remote forwarding.

Explanation: A remote forwarding request has failed.

System action: The program ends.

User response: Check for additional error messages displayed with this message, and take appropriate action. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the error messages displayed with this message. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1282 Bad dynamic forwarding specification '*value***'**

Explanation: The **ssh** –D option is set to a bad value *value*.

System action: The program ends.

User response: Verify that the value for the **ssh** –D option is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** –D option. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1283 Master running (pid=*pid***)**

Explanation: The master process of the specified multiplexed connection is running. Its process id is *pid*.

System action: The program ends.

User response: No response required. This message reports information on a multiplexed connection.

FOTS1284 Exit request sent.

Explanation: An exit request was sent to the master process of the specified multiplexed connection.

System action: The program ends.

User response: No response required. This message reports information on a multiplexed connection.

FOTS1285 Shared connection to *host_name* **closed.**

Explanation: The shared connection to the master process of the specified multiplexed connection on host *host_name* has been closed.

System action: The program ends.

User response: No response required. This message reports information on a multiplexed connection.

FOTS1287 Warning: Identity file *filename* **does not exist.**

Explanation: The filename specified with the ssh -i option does not exist.

System action: The program continues.

User response: Verify that the filename specified is correct and exists.

FOTS1288 no support for smartcards.

Explanation: ssh on z/OS does not provide support for smart cards.

System action: The program continues.

User response: Reissue the command without the smart card option (-I for ssh).

System programmer response: None.

FOTS1289 No support for Kerberos ticket or AFS token forwarding.

Explanation: ssh on z/OS does not provide support for Kerberos tickets or AFS tokens.

System action: The program continues.

User response: Reissue the command without the option to disable Kerberos ticket and AFS token forwarding (-k for ssh).

System programmer response: None.

FOTS1290 Bad escape character '*escape char***'.**

Explanation: You specified an invalid escape character.

System action: The program ends.

User response: An escape character can be either a single character or a control character. Reissue the command with a valid escape character.

System programmer response: None.

FOTS1291 Unknown cipher type '*cipher_spec***'**

Explanation: ssh does not recognize the cipher specified with the -c option.

System action: The program ends.

User response: Check ssh documentation for a valid cipher specification.

System programmer response: None.

FOTS1292 Unknown mac type '*mac_spec***'**

Explanation: ssh does not recognize the message authentication code specified with the -m option.

System action: The program ends.

User response: Check ssh documentation for a valid MAC specification.

System programmer response: None.

FOTS1293 Bad port '*port***'**

Explanation: The port number specified is invalid. It should be greater than zero and less than or equal to 65535.

System action: The program ends.

User response: Reissue **ssh** with a valid port number.

System programmer response: None.

FOTS1294 Bad forwarding port(s) '*port***'**

Explanation: One of the port numbers specified with ssh options -R or -L are invalid. A port number should be greater than zero and less than or equal to 65535.

System action: The program ends.

User response: Reissue **ssh** with valid port numbers.

System programmer response: None.

FOTS1295 Bad forwarding specification '*specification***'**

Explanation: The syntax of specification is incorrect.

System action: If the forwarding specification was issued through an opened command line (through an escape character), the program continues. Otherwise, the program ends.

User response: Check ssh documentation for the proper syntax.

System programmer response: None.

FOTS1296 Bad dynamic port '*port***'**

Explanation: The port number specified is invalid. It should be greater than zero and less than or equal to 65535.

System action: The program ends.

User response: Reissue **ssh** with a valid port number.

System programmer response: None.

FOTS1297 You must specify a subsystem to invoke.

Explanation: You specified **ssh** -s without a subsystem.

System action: The program ends.

User response: Reissue **ssh** -s with a subsystem as the command.

FOTS1298 rresvport: af=*family system_error*

Explanation: An error occurred while ssh was attempting to connect to a privileged port (because configuration option UsePrivilegedPort was specified). A call to bind(), socket(), or getsockname() may have

failed, or the address family *family* is not supported. The system error is displayed with this message.

System action: The program continues.

User response: Check that ssh is setuid root. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1299 socket: *system error*

Explanation: A call to socket() failed. The system error is displayed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1301 getaddrinfo: *bindaddress*: *system error*

Explanation: The ssh client failed when trying to get the address information for the interface specified by ssh configuration option BindAddress. The system error is displayed with this message.

System action: The program continues.

User response: Verify *bindaddress* is valid.

FOTS1302 bind: *bindaddress*: *system error*

Explanation: A call to bind() failed with the *bind address* specified by ssh configuration option BindAddress.

System action: The program continues.

User response: Verify *bindaddress* is valid.

FOTS1303 ssh_connect: getnameinfo failed

Explanation: ssh was unable to get the name information from an IP address.

System action: The program continues.

User response: Check that all the specified addresses for the host are valid.

FOTS1304 setsockopt SO_KEEPALIVE: *system error*

Explanation: The KeepAlive configuration option was specified but the setsockopt() system call for SO_KEEPALIVE failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1305 No *key type* **host key is known for** *hostname* **and you have requested strict checking.**

Explanation: While ssh is checking if a host key is valid, it could not find a key for *hostname*.

System action: The program ends.

User response: Check that the file containing the list of known hosts exists. Check that the key for the desired host is in the known hosts file.

System programmer response: None.

FOTS1306 Keyboard-interactive authentication is disabled to avoid man-in-the-middle attacks.

Explanation: Strict host key checking has been requested, so keyboard-interactive authentication has been disabled to prevent man-in-the-middle attacks. Challenge-response authentication is also disabled.

System action: The program continues.

User response: Check that the host key in the user's known hosts file is valid.

FOTS1307 Challenge/response authentication is disabled to avoid man-in-the-middle attacks.

Explanation: Strict host key checking has been requested, so challenge-response authentication has been disabled to prevent man-in-the-middle attacks.

System action: The program continues.

User response: Check that the host key in the user's known hosts file is valid.

FOTS1308 @ WARNING: POSSIBLE DNS SPOOFING DETECTED! @ The *type* **host key for** *hostname* **has changed, and the key for the according IP address** *ip address problem*. **This could either mean that DNS SPOOFING is happening or the IP address for the host and its host key have changed at the same time.**

Explanation: See message text.

System action: The program continues unless strict host key checking is enabled.

User response: Check whether the host key is accurate.

FOTS1314 Offending key for IP in *filename*:*line_number*

Explanation: The key found on line *line_number* of file *filename* is not valid. The host's public key may have changed.

System action: The program continues unless strict host key checking is enabled.

User response: Check the specified line number and file for a valid host key.

FOTS1315 Update the SSHFP RR in DNS with the new host key to get rid of this message.

Explanation: The SSH fingerprint resource record in DNS does not have the proper data for the host key.

System action: The program continues.

User response: Contact your system administrator to fix the resource record.

System programmer response: Update the DNS server to correct the problem.

FOTS1316 Bogus return (*return code*) from select()

Explanation: A call to select() failed with return code *return code*.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1317 @ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @ IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY! Someone could be eavesdropping on you right now (man–in–the–middle attack)! It is also possible that the *keytype* host key has just been changed. The fingerprint for the *keytype* key sent by the remote host is *fingerprint*. Please contact your system administrator. Add correct host key in *userhostfile* to get rid of this message. Offending key in *hostfile*:*line_number*

Explanation: ssh has detected that the remote host key has changed.

System action: The program continues unless strict host key checking is enabled.

User response: Check that you have a valid host key for the remote host.

FOTS1325 *key type* host key for *host name* has changed and you have requested strict checking.

Explanation: Strict host key checking (ssh configuration option StrictHostKeyChecking) is enabled which causes ssh to exit if the host key has changed.

System action: The program ends.

User response: Edit the key in your user known hosts file.

System programmer response: None.

FOTS1326 Password authentication is disabled to avoid man–in–the–middle attacks.

Explanation: Strict host key checking (ssh configuration option StrictHostKeyChecking) has not been requested, so the connection is allowed, but password authentication is disabled.

System action: The program continues.

User response: Check that the host key in the user's known hosts file is valid.

System programmer response: None.

FOTS1327 Agent forwarding is disabled to avoid man-in-the-middle attacks.

Explanation: Strict host key checking (ssh configuration option StrictHostKeyChecking) has not been requested, so the connection is allowed, but agent forwarding is disabled.

System action: The program continues.

User response: Check that the host key in the user's known hosts file is valid.

System programmer response: None.

FOTS1328 X11 forwarding is disabled to avoid man-in-the-middle attacks.

Explanation: Strict host key checking (ssh configuration option StrictHostKeyChecking) has not been requested, so the connection is allowed, but X11 forwarding is disabled.

System action: The program continues.

User response: Check that the host key in the user's known hosts file is valid.

System programmer response: None.

FOTS1329 Port forwarding is disabled to avoid man-in-the-middle attacks.

Explanation: Strict host key checking (ssh configuration option StrictHostKeyChecking) has not

been requested, so the connection is allowed, but port forwarding is disabled.

System action: The program continues.

User response: Check that the host key in the user's known hosts file is valid.

System programmer response: None

FOTS1330 Exiting, you have requested strict checking.

Explanation: Strict host key checking (ssh configuration option StrictHostKeyChecking) has been requested, CheckHostIp was enabled, and the host name is not known.

System action: The program ends.

User response: Make sure the host key for the remote host is in the user's known hosts file.

System programmer response: None.

FOTS1331 dup2 stdin

Explanation: A call to dup2() failed. The system error is displayed with this message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1332 dup2 stdout

Explanation: A call to dup2() failed. The system error is displayed with this message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1333 *shell_path* **:** *message*

Explanation: A call to execv() failed to execute *shell_path*. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1334 Could not create pipes to communicate with the proxy: *system error*

Explanation: A call to pipe() failed. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1335 fork failed: *error_message*

Explanation: The fork() system call failed. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1336 *program name***: Could not resolve hostname** *host***:** *system error*

Explanation: The ssh client failed when trying to get the address information for *host* The system error is displayed with this message.

System action: The program ends.

User response: Verify *host* is valid.

FOTS1337 ssh_exchange_identification: read: *system error*

Explanation: ssh was unable to read the other side of the connection's identification information. A read() on the socket failed. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1338 ssh_exchange_identification: Connection closed by remote host

Explanation: While attempting to read the other side of the connection's version identification, the connection was closed by the remote host.

System action: The program ends.

User response: Verify that the remote host is still

operable. Verify that the remote host has an implementation of SSH which is compatible with OpenSSH.

FOTS1339 Bad remote protocol version identification: '*server version string***'**

Explanation: The OpenSSH version of the server does not match the version of the client.

System action: The program ends.

User response: Check that the local and remote versions of OpenSSH are compatible.

System programmer response: None.

FOTS1340 Remote machine has too old SSH software version.

Explanation: The remote sshd minor version is less than 3.

System action: The program ends.

User response: Verify local OpenSSH suite is compatible with remote version.

FOTS1341 Protocol major versions differ: *localprotocol* **vs.** *remoteprotocol*

Explanation: The ssh client requested using SSH Protocol Version *localprotocol*, but the remote server requires *remoteprotocol*.

System action: The program ends.

User response: Reissue **ssh** using the protocol that the server expects, or contact system administrator of remote machine.

FOTS1342 write: *system error*

Explanation: A call to write() failed for the outgoing socket. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1343 check_host_key: getnameinfo failed

Explanation: ssh was unable to get the name information for the current host.

System action: The program ends.

User response: Check that all the specified addresses for the host are valid.

FOTS1344 internal error

Explanation: An internal error has occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1345 Bad passphrase.

Explanation: During RSA authentication for protocol version 1, the given passphrase is invalid for the current RSA key.

System action: The program continues.

User response: Verify you entered the correct passphrase.

FOTS1346 Permission denied, please try again.

Explanation: You do not have permission to log into the system.

System action: The program continues.

User response: Contact system administrator for the system in which you are refused access.

FOTS1348 try_agent_authentication: BN_new failed

Explanation: The ssh client tried to authenticate using the ssh-agent. A call to the OpenSSL function BN_new() failed. BN_new() allocates and initializes a BIGNUM structure. An internal error has occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1349 try_rsa_authentication: BN_new failed

Explanation: The ssh client tried to authenticate using RSA authentication. A call to the OpenSSL function BN_new() failed. BN_new() allocates and initializes a BIGNUM structure. An internal error has occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1350 try_rhosts_rsa_authentication: BN_new failed

Explanation: The ssh client tried to authenticate using combined rhosts or /etc/hosts.equiv authentication and RSA authentication. A call to the OpenSSL function BN_new() failed. BN_new() allocates and initializes a

BIGNUM structure. An internal error has occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1351 Kerberos v4: Malformed response from server

Explanation: The ssh client got an invalid response from the server.

System action: The program ends.

User response: Verify Kerberos is configured properly. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1352 Host key verification failed.

Explanation: During SSH key exchange, ssh was unable to verify the host key.

System action: The program continues.

User response: Verify your list of known hosts is accurate. Check if the remote host changed their host key.

FOTS1353 ssh_kex: BN_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1354 respond_to_rsa_challenge: host_key *hostbits* **< server_key** *serverbits* **+ SSH_KEY_BITS_RESERVED** *bits*

Explanation: SSH Protocol Version 1 key exchange failed because the difference between the number of bits in the host's public key and the number of bits of the server key was not greater than *bits*. The host key length and server key length need to differ by at least *bits* bits.

System action: The program ends.

User response: Try a different authentication method.

FOTS1355 respond_to_rsa_challenge: server_key *serverbits* **< host_key** *hostbits* **+ SSH_KEY_BITS_RESERVED** *bits*

Explanation: SSH Protocol Version 1 key exchange failed because the difference between the number of bits in the host's public key and the number of bits of the server key was not greater than *bits*. The host key length and server key length need to differ by at least *bits* bits.

System action: The program ends.

User response: Try a different authentication method.

FOTS1356 Selected cipher type *cipher* **not supported by server.**

Explanation: The cipher *cipher* is not supported by the remote sshd. Note that cipher "des" is not supported by IBM z/OS sshd.

System action: The program ends.

User response: Reissue **ssh** client with a remotely-supported cipher.

FOTS1357 ssh_userauth1: server supports no auth methods

Explanation: The server doesn't support any authentication methods for SSH Protocol Version 1.

System action: The program ends.

User response: Try using Protocol Version 2.

FOTS1358 Permission denied.

Explanation: All authentication methods have failed.

System action: The program ends.

User response: Verify your setup is correct.

FOTS1359 input_userauth_pk_ok: type mismatch for decoded key (received *keytype*, **expected** *keytype2***)**

Explanation: The key from across the network claimed to be a key of type *keytype2*, but the decoded key was actually key type *keytype*.

System action: The program continues.

User response: Check that your public key on the remote host is correct.

FOTS1361 ssh_keysign: no installed: *system error*

Explanation: Could not stat() /usr/lib/ssh/ssh-keysign.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1362 ssh_keysign: fflush: *system error*

Explanation: A call to fflush() failed for stdout. The system error is displayed with this message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1363 ssh_keysign: pipe: *system error*

Explanation: A call to pipe() failed for stdout. The system error is displayed with this message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1364 ssh_keysign: couldn't send request

Explanation: The ssh client could not successfully send a message to ssh-keysign.

System action: The program ends.

User response: Verify that ssh-keysign exists. Verify your setup is correct. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1365 ssh_keysign: fork: *system error*

Explanation: A call to fork() failed for stdout. The system error is displayed with this message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1366 ssh_keysign: no reply

Explanation: The ssh client did not receive a response from ssh-keysign.

System action: The program continues.

User response: Verify that ssh-keysign exists. Verify your setup is correct. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1367 ssh_keysign: bad version

Explanation: The version of ssh-keysign does not match that of the ssh client.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Verify that the ssh-keysign and ssh clients installed are those provided by IBM. Follow local procedures for reporting problems to IBM.

FOTS1368 userauth_hostbased: cannot get local ipaddr/name

Explanation: During hostbased authentication, ssh could not find a name for the local host.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Verify that the DNS setup on the local system is correct. Follow local procedures for reporting problems to IBM.

FOTS1369 key_sign failed

Explanation: The ssh client was unable to authenticate using RSA-based host authentication because ssh-keysign failed.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Verify that ssh-keysign exists. Verify that the setup is correct. Follow local procedures for reporting problems to IBM.

FOTS1370 Host key verification failed.

Explanation: The ssh client was unable to authenticate using hostbased authentication because it could not verify the host key.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Verify that the SSH setup is correct. Follow local procedures for reporting problems to IBM.

FOTS1371 denied SSH2_MSG_SERVICE_ACCEPT: *type*

Explanation: During user authentication, ssh expected a packet of type SSH2_MSG_SERVICE_ACCEPT but instead received one of type *type*.

System action: The program ends.

User response: Verify that the remote server is working properly. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1372 ssh_userauth2: internal error: cannot send userauth none request

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1373 Permission denied (*authentication_list*).

Explanation: You were refused access to the system after all the authentication methods in *authentication_list* were attempted.

System action: The program ends.

User response: Verify you typed your password and/or passphrase correctly. Verify with remote system security administrator whether or not they intended you have access. Your user may be listed as part of DenyUsers or DenyGroups on the remote server.

System programmer response: None.

FOTS1374 input_userauth_error: bad message during authentication: type *type*

Explanation: During user authentication, ssh received a packet type it did not expect.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1375 input_userauth_success: no authentication context

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1376 input_userauth_failure: no authentication context

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1377 input_userauth_pk_ok: no authentication context

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1378 input_userauth_passwd_changereq: no authentication context

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1379 userauth_pubkey: internal error

Explanation: An internal error has occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1380 input_userauth_info_req: no authentication context

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1381 **ssh_keysign: dup2:** *system error*

Explanation: A call to dup2() failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1382 **Server denied authentication request:** *type*

Explanation: During user authentication, ssh expected a packet of type SSH2_MSG_SERVICE_ACCEPT but instead received one of type *type*.

System action: The program ends.

User response: Verify that the remote server is working properly. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1383 **ssh_keysign: exec(***keysignpath***):** *system error*

Explanation: A call to exec() failed when trying to execute ssh-keysign.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1385 **tcsetattr**

Explanation: A call to tcsetattr() failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1386 **tcgetattr**

Explanation: A call to tcgetattr() failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1388 *filename***: line** *line number***: Bad configuration option:** *configuration option*

Explanation: An option specified in an ssh configuration file is invalid.

System action: The program ends.

User response: Check *line number* of the ssh configuration file *filename* for the invalid option.

System programmer response: None.

FOTS1389 **Privileged ports can only be forwarded by root.**

Explanation: While ssh was attempting to add a locally forwarded port, the port number specified is privileged but the user isn't authorized to use a privileged port.

System action: The program ends.

User response: Reissue the ssh command with a valid port (either in ssh configuration file or on command line.)

System programmer response: None.

FOTS1390 **Too many local forwards (max** *max forwards***).**

Explanation: The user attempted to specify more local forwards than are allowed by ssh. ssh currently allows *max forwards*.

System action: The program ends.

User response: Reissue **ssh** without a locally forwarded port.

System programmer response: None.

FOTS1391 **Too many remote forwards (max** *max_forwards***).**

Explanation: The user attempted to specify more remote forwards than are allowed by ssh. ssh currently allows a maximum of *max_forwards*.

System action: The program ends.

User response: Reissue **ssh** without a remotely forwarded port.

System programmer response: None.

FOTS1392 *filename* **line** *line number***: Missing yes/no argument.**

Explanation: While parsing the configuration file *filename*, ssh expected a yes/no argument but it is missing.

System action: The program ends.

User response: Check the specified line number in the file for syntax errors. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1393 *filename* **line** *line number*: **Bad yes/no argument.**

Explanation: While parsing the configuration file *filename*, ssh expected a yes/no argument but instead encountered a syntax error.

System action: The program ends.

User response: Check the specified line number in the file for syntax errors. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1394 *filename* **line** *line number*: **Missing yes/no/ask argument.**

Explanation: While parsing the configuration file *filename*, ssh expected a yes/no/ask argument with the StrictHostKeyChecking option, but it is missing.

System action: The program ends.

User response: Check the specified line number in the file for syntax errors. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1395 *filename* **line** *line number*: **Bad yes/no/ask argument.**

Explanation: While parsing the configuration file *filename*, ssh expected a yes/no/ask argument with the StrictHostKeyChecking option, but instead encountered a syntax error.

System action: The program ends.

User response: Check the specified line number in the file for syntax errors. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1396 *filename* **line** *line_number*: **Missing argument.**

Explanation: A ssh_config keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

User response: Verify that a value for the ssh_config keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config keywords. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1397 *filename* **line** *line number*: **Too many identity files specified (max** *max*).

Explanation: The maximum number of authentication identity files and key ring certificates (*max*) that can be specified in configuration files or command line has been exceeded.

System action: The program ends.

User response: Reissue the command with a smaller number of identity files or key ring certificates. Check the number of times the IdentityFile or IdentityKeyRingLabel configuration options were specified in the configuration files.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1398 *filename* **line** *line number*: **missing time value.**

Explanation: The ssh configuration file *filename* or command line has a configuration option which expects a time value, but the corresponding time value is missing. Options which expect time values include ConnectTimeout.

System action: The program ends.

User response: Check *line number* of the ssh configuration file *filename* for the failing option, add a time value and reissue ssh.

FOTS1399 *filename* **line** *line number*: **invalid time value.**

Explanation: The ssh configuration file *filename* has a configuration option which expects a time value, but the corresponding time value is invalid. Options which expect time values include ConnectTimeout.

System action: The program ends.

User response: Check *line number* of the ssh configuration file *filename* for the failing option, correct the time value and reissue **sshd**.

FOTS1401 *filename* **line** *line number*: **Bad number "***number***"**

Explanation: While parsing *filename*, ssh encountered an invalid number.

- With option NumberOfPasswordPrompts or ConnectionAttempts, *number* must be an integer between 0 and 2147483647(LONG_MAX).
- With option CompressionLevel, *number* must be an integer between 1 and 9.
- With option Port, *number* must be an integer between 1 and 65535(USHRT_MAX).

System action: The program ends.

User response: Check the specified line number in the file to make sure number is valid. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1404 *filename* **line** *line number*: **Bad cipher '***cipher***'.**

Explanation: While parsing *filename*, ssh encountered an invalid *cipher* after the Cipher option.

System action: The program ends.

User response: Check the specified line number in the file to make sure the cipher is valid. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1405 **Unsupported AddressFamily "***argument***"**

Explanation: The argument supplied with the ssh configuration option AddressFamily is invalid. Valid arguments include "inet", "inet6", or "any".

System action: The program ends.

User response: Reissue the command with a valid value for AddressFamily.

FOTS1406 *filename* **line** *line number*: **Bad SSH2 cipher spec '***ciphers***'.**

Explanation: While parsing *filename*, ssh encountered invalid *ciphers* after the Ciphers option.

System action: The program ends.

User response: Check the specified line number in the file to make sure ciphers are valid. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1407 *filename* **line** *line number*: **Unsupported option "***keyword***"**

Explanation: The ssh configuration option *keyword* is not supported.

System action: The program continues.

User response: Remove the unsupported option from the specified line in the ssh configuration file *filename*.

FOTS1408 *filename* **line** *line number*: **Bad SSH2 Mac spec '***MAC algorithms***'.**

Explanation: While parsing *filename*, ssh encountered invalid *MAC algorithms* after the MACs option.

System action: The program ends.

User response: Check the specified line number in the file to make sure the *MAC algorithms* are valid. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1410 *filename* **line** *line number*: **Bad protocol 2 host key algorithms '***algorithms***'.**

Explanation: While parsing *filename*, ssh encountered invalid protocol 2 host key algorithms after the HostKeyAlgorithms option.

System action: The program ends.

User response: Check the specified line number in the file to make sure the protocol 2 host key *algorithms* are valid. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1412 *filename* **line** *line number*: **Bad protocol spec '***protocol***'.**

Explanation: While parsing *filename*, ssh encountered an invalid *protocol* version after the Protocol option.

System action: The program ends.

User response: Check the specified line number in the file to make sure you have a valid protocol version. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1413 *filename* **line** *line number*: **unsupported log level '***level***'**

Explanation: While parsing *filename*, ssh encountered an invalid log *level* after the LogLevel option.

System action: The program ends.

User response: Check the specified line number in the file to make sure you have a valid log level. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified

line number in the file for syntax errors.

FOTS1414 *filename* **line** *line number*: **Missing port argument.**

Explanation: While parsing *filename*, ssh encountered a syntax error for a configuration option. The configuration option requires an argument after the keyword.

System action: The program ends.

User response: Check the specified line number in the file for syntax errors. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1415 *filename* **line** *lineno*: **Bad listen port.**

Explanation: While parsing *filename*, ssh encountered an invalid argument for either the LocalForward or RemoteForward configuration option.

System action: The program ends.

User response: Check the specified line number in the file to make sure you have a valid argument for the configuration option in error. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1416 *filename* **line** *lineno*: **Missing target argument.**

Explanation: While parsing *filename*, the target argument for either the LocalForward or RemoteForward configuration option is missing.

System action: The program ends.

User response: Check the specified line number in the file to make sure you have a valid argument for the configuration option in error. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1417 *filename* **line** *lineno*: **Bad forwarding specification.**

Explanation: While parsing *filename*, ssh encountered an invalid argument for either the LocalForward, RemoteForward or DynamicForward configuration option.

System action: The program ends.

User response: Check the specified line number in the file to make sure you have a valid argument for the configuration option in error. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1418 *filename* **line** *lineno*: **Bad forwarding port.**

Explanation: One of the port numbers specified with ssh configuration options LocalForward or RemoteForward is invalid. A port number should be greater than zero and less than or equal to 65535.

System action: The program ends.

User response: Check the specified line number in the file to make sure you have a valid argument for the configuration option in error. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1420 *filename* **line** *lineno*: **Badly formatted port number.**

Explanation: While parsing *filename*, ssh encountered an invalid argument for the DynamicForward configuration option.

System action: The program ends.

User response: Check the specified line number in the file to make sure you have a valid argument for the configuration option in error. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1422 *filename* **line** *line number*: **Bad escape character.**

Explanation: You specified an invalid escape character in the ssh configuration file.

System action: The program ends.

User response: An escape character can be either a single character or a control character. Reissue the command with a valid escape character.

System programmer response: None

FOTS1423 **process_config_line: Unimplemented opcode** *opcode*

Explanation: An internal error has occurred.

System action: The program ends.

User response: Contact your system administrator to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1424 *filename* **line** *line number*: **garbage at end of line; "***text***".**

Explanation: The extra text *text* was found after a configuration option. Please check the specified filename.

System action: The program ends.

User response: Check the specified line number in the file for syntax errors. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1425 *filename*: **terminating,** *options* **bad configuration options**

Explanation: ssh has encountered at least one invalid configuration option.

System action: The program ends.

User response: Check the specified filename for syntax errors. Contact your system administrator if the configuration file is global.

System programmer response: Check the specified line number in the file for syntax errors.

FOTS1426 **fork:** *system error*

Explanation: A call to fork() failed. The system error is displayed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1427 **client_channel_closed: id** *id1* **!= session_ident** *id2*

Explanation: The ssh client is closing a channel with *id1* but the current session id is *id2*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1428 **Write failed flushing stdout buffer.**

Explanation: A call to write() failed when attempting to write to stdout.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1429 **Write failed flushing stderr buffer.**

Explanation: A call to write() failed when attempting to write to stderr.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1430 **Warning: ssh server tried X11 forwarding.**

Explanation: The ssh configuration option ForwardX11 was disabled but the server requested an X11 channel.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for handling security problems.

FOTS1431 **Warning: this is probably a break–in attempt by a malicious server.**

Explanation: The ssh client detected the server attempting to bypass some ssh setup. This error message is usually displayed with another message describing what ssh sees in error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for handling security problems.

FOTS1432 **Warning: ssh server tried agent forwarding.**

Explanation: The ssh configuration option ForwardAgent was disabled but the server requested an X11 channel.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for handling security problems.

FOTS1434 **client_input_channel_req: no channel** *session channel identifier*

Explanation: The server wanted to request a new channel, but no session channel exists for the client.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local

procedures for reporting problems to IBM.

FOTS1435 **client_input_channel_req: channel** *session channel identifier***: wrong channel:** *requested channel*

Explanation: The server wanted to request a new channel, but the channel requested by the server doesn't match that of the client's session.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1436 **client_input_channel_req: channel** *requested channel***: unknown channel**

Explanation: The channel identifier sent by the server is not recognized by the client.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1437 **Killed by signal** *signal number***.**

Explanation: The ssh client was killed by signal *signal number*.

System action: The program ends.

User response: Determine what caused a signal to be sent to your process.

System programmer response: None.

FOTS1438 **Could not load host key:** *host key file*

Explanation: The file *host key file* could not be loaded. The file may not exist or is not readable. The permissions on the file may be incorrect. The passphrase may have been entered incorrectly.

System action: The program continues.

User response: Check that *host key file* exists and has the proper permissions. Verify that the correct passphrase was used.

System programmer response: None.

FOTS1439 **getnameinfo failed:** *system error*

Explanation: ssh was unable to get the name information for the current host.

System action: The program continues.

System programmer response: Check that all the specified addresses for the host are valid.

FOTS1440 **listen_sock O_NONBLOCK:** *system error*

Explanation: A call to fcntl() to set O_NONBLOCK failed for the listening socket.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1441 **setsockopt SO_REUSEADDR:** *system error*

Explanation: A call to setsockopt() to set SO_REUSEADDR failed for the listening socket. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1442 **Bind to port** *port* **on** *host* **failed:** *system error*

Explanation: sshd was unable to bind the socket to the desired port. A call to bind() failed and the system error is displayed.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1443 **select:** *system error*

Explanation: sshd is waiting in a select() call until there is a connection. This call to select() failed. The system error is displayed.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1444 **accept:** *system error*

Explanation: A call to accept() failed. The system error is displayed.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1445 **newsock del O_NONBLOCK:** *system error*

Explanation: A call to fcntl() failed. The system error is displayed.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1446 **__poe() failed for accepted socket:** *system error*

Explanation: A call to __poe() failed. The system error is displayed.

System action: The daemon handling the connection ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1447 **setsid:** *system error*

Explanation: While sshd was attempting to create a new session and process group, a call to setsid() failed. The system error is displayed.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1448 **setsockopt SO_KEEPALIVE:** *system error*

Explanation: A call to setsockopt() to set SO_KEEPALIVE failed for the listening socket. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1449 **do_ssh1_kex: bad session key len from** *remote_ip*: **session_key_int** *length* **> sizeof(session_key)** *session_key_length*

Explanation: During key exchange, the remote host's session key (*length*) is larger than what this daemon supports (*session_key_length*).

System action: The program continues.

User response: Follow local procedures for reporting problems to IBM.

FOTS1450 **Timeout before authentication for** *remote_ip*

Explanation: sshd timed-out before the user authenticated itself. The sshd administrator may have configured too low a value for the login grace time. The sshd -g option or sshd_config keyword LoginGraceTime controls this value.

System action: The program ends.

System programmer response: Follow local procedures for handling user authentication timeouts.

FOTS1451 **Privilege separation user** *user_name* **does not exist**

Explanation: The user *user_name* must exist when privilege separation is enabled via the sshd_config UsePrivilegeSeparation keyword.

System action: The program ends.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on privilege separation setup and the sshd_config UsePrivilegeSeparation keyword.

FOTS1452 **chroot("***chroot_dir***"):** *system error*

Explanation: sshd attempted to chroot() to *chroot_dir*, which is the chroot directory used by sshd during privilege separation.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1453 **chdir("/"):** *system error*

Explanation: sshd failed while attempting to chdir() to "/". The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1454 **setgid failed for** *groupid*

Explanation: A call to setgid() failed for the privilege separation user's group id.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1455 **setgroups:** *system error*

Explanation: A call to setgroups() failed for the privilege separation user's group id. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1456 **fork of unprivileged child failed:** *system error*

Explanation: While sshd was attempting to set up the unprivileged child process, a call to fork() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1457 **TCP/IP TERMINATED. Will attempt to restart every** *seconds* **seconds.**

Explanation: TCP/IP has gone down or has not been started yet. sshd will sleep for *seconds* seconds, and try again. This message will only be displayed once, not for each restart attempt.

System action: The program continues.

System programmer response: Wait until sshd recognizes the new stack.

FOTS1458 **setibmsockopt SO_EioIfNewTP :** *error_code*

Explanation: The setibmsockopt() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1459 **Missing privilege separation directory:** *chroot_dir*

Explanation: The directory used by sshd during privilege separation is missing or is not a directory.

System action: The program ends.

System programmer response: Check that *chroot_dir* exists and is a directory. It should also be owned by uid 0, and not be group or world-writable.

FOTS1460 **Bad owner or mode for** *chroot_dir*

Explanation: The directory used by sshd during privilege separation is not owned by uid 0 or is group or world-writable.

System action: The program ends.

System programmer response: *chroot_dir* should also be owned by uid 0, and not be group or world-writable.

FOTS1461 **Couldn't create pid file "***filename***":** *system error*

Explanation: The sshd pid file *filename* could not be opened. A call to fopen() failed when attempting to open the file. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1462 **Too many listen sockets. Enlarge MAX_LISTEN_SOCKS**

Explanation: The number of sockets for which sshd is attempting to listen is greater than what it can currently handle. The current value is 16.

System action: The program ends.

System programmer response: Verify less than 16 addresses are specified with configuration option ListenAddress.

FOTS1463 **listen:** *system error*

Explanation: sshd attempted to listen on a port, and a call to listen() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Check the log information for the failing port number. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1464 **Cannot bind any address.**

Explanation: sshd was not able to bind to any of the addresses listed by configuration option ListenAddress.

System action: The program ends.

System programmer response: Check sshd log output for specific bind failures.

FOTS1465 *directory* **must be owned by root and not group or world-writable.**

Explanation: The chroot directory *directory* used by sshd during privilege separation is either not owned by root, or is group or world-writable.

System action: The program ends.

System programmer response: Check the permissions and ownership of the directory.

FOTS1466 **do_connection:** *remote_ip*: **server_key** *server_num_bits* **<** **host_key** *host_num_bits* **+ SSH_KEY_BITS_RESERVED** *ssh_key_bits_reserved*

Explanation: The host key length *host_num_bits* and the server key length *server_num_bits* should differ by the number of bits specified by *ssh_key_bits_reserved*.

System action: The program ends.

System programmer response: Invoke **sshd** (using the -b option) with a larger number of bits for the server key.

FOTS1467 **do_connection:** *remote_ip*: **host_key** *host_num_bits* **<** **server_key** *server_num_bits* **+ SSH_KEY_BITS_RESERVED** *ssh_key_bits_reserved*

Explanation: The host key length *host_num_bits* and the server key length *server_num_bits* should differ by the number of bits specified by *ssh_key_bits_reserved*.

System action: The program ends.

System programmer response: Make the host key and the server key conform to this property.

FOTS1468 **do_ssh1_kex: BN_new failed**

Explanation: During key exchange, a call to the OpenSSL function BN_new() failed. An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for handling user authentication timeouts.

FOTS1487 **TCP/IP TERMINATED, or new stack started.**

Explanation: sshd has received an error which is interpreted as indicating that TCP/IP has terminated or that a new stack has been started. This message is preceded by one or more other messages indicating what error was received. Typically, a call to accept() will have failed with a system error of EIO.

System action: The program continues. sshd attempts to reinitialize the sockets for the services in the configuration file. If that fails, sshd attempts to reinitialize the sockets in repeated intervals.

System programmer response: Wait until sshd recognizes a new TCP/IP stack.

FOTS1488 **too many ports.**

Explanation: The **sshd** -p option was specified more times than it can handle. The maximum number of ports allowed by sshd is 256.

System action: The program ends.

System programmer response: Reissue **sshd** with a valid number of ports.

FOTS1489 **Bad port number.**

Explanation: The port number specified with sshd -p is invalid. It should be a number greater than 0 and less than or equal to 65535.

System action: The program ends.

System programmer response: Reissue **sshd** with a valid port number.

FOTS1490 **Invalid login grace time.**

Explanation: The login grace time specified with sshd –g is invalid.

System action: The program ends.

System programmer response: See *IBM Ported Tools for z/OS User's Guide* for more information on sshd –g.

FOTS1491 **Invalid key regeneration interval.**

Explanation: The key regeneration interval specified with sshd –k is invalid.

System action: The program ends.

System programmer response: See *IBM Ported Tools for z/OS User's Guide* for more information on sshd –k.

FOTS1492 **too many host keys.**

Explanation: The maximum number of host key files and host key ring certificates that can be specified in configuration files or the command line has been exceeded.

System action: The program ends.

System programmer response: Reissue **sshd** with a smaller number of host keys. See *IBM Ported Tools for z/OS User's Guide* for more information on the maximum allowed.

FOTS1493 Invalid utmp length.

Explanation: The length specified with sshd -u is larger than what can be stored in the utmpx database.

System action: The program ends.

System programmer response: Reissue **sshd** with a smaller value for the -u option.

FOTS1494 Extra argument *argument***.**

Explanation: sshd was specified with too many arguments.

System action: The program ends.

System programmer response: Reissue **sshd** with the proper syntax.

FOTS1495 Bad server key size.

Explanation: The number of bits specified for the server key is invalid. The server key bits (controlled by configuration option ServerKeyBits) must be between 512 and 32768 inclusive.

System action: The program ends.

System programmer response: Reissue **sshd** with a valid number of bits for the server key.

FOTS1496 do_authloop: BN_new failed

Explanation: During RSA authentication in sshd, a call to the OpenSSL function BN_new() failed. An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1497 INTERNAL ERROR: authenticated invalid user *username*

Explanation: The user *username* is not a valid user, but was successfully authenticated.

System action: The program ends.

System programmer response: Follow local procedures for handling security problems.

FOTS1498 Port of Entry information not retained. uname() failed : *system error*

Explanation: A call to uname() failed. If there is a system error, it is displayed. Because of this failure, the port of entry information has not been retained. Access to the system by the attempting user may fail.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1499 Port of Entry information not retained. strtol() failed : *system error*

Explanation: A call to strtol() failed. If there is a system error, it is displayed with this message. Because of this failure, the port of entry information has not been retained. Access to the system by the attempting user may fail.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1501 input_userauth_request: no authctxt

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1502 INTERNAL ERROR: authenticated invalid user *user*

Explanation: The user *username* is not a valid user, but was successfully authenticated.

System action: The program ends.

System programmer response: Follow local procedures for handling security problems.

FOTS1503 __passwd: *system error*

Explanation: A call to __passwd() failed. The system error is displayed with this message.

System action: The program continues.

User response: Check that you entered the right password. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1504 userauth_hostbased: cannot decode key: *keytype*

Explanation: During hostbased authentication, sshd was unable to decode the public key of type *keytype* which was sent from across the network.

System action: The program continues.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1505 **userauth_hostbased: type mismatch for decoded key (received** *keytype1***, expected** *keytype2***)**

Explanation: The key sshd received across the network declared it's type to be *keytype2*, but was actually *keytype1* when decoded.

System action: The program continues.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1506 **userauth_pubkey: cannot decode key:** *keytype*

Explanation: During public key authentication, sshd was unable to decode the public key of type *keytype* which was sent from across the network.

System action: The program continues.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1507 **userauth_pubkey: type mismatch for decoded key (received** *keytype1***, expected** *keytype2***)**

Explanation: The key sshd received across the network declared it's type to be *keytype2*, but was actually *keytype1* when decoded.

System action: The program continues.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1508 **get_challenge: numprompts < 1**

Explanation: Challenge-response authentication failed because the number of prompts to the user was exceeded.

System action: The program ends.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1509 **input_userauth_info_response: no authctxt**

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1510 **input_userauth_info_response: no kbdintctxt**

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1511 **input_userauth_info_response: no device**

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1512 **input_userauth_info_response: wrong number of replies**

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1513 **input_userauth_info_response: too many replies**

Explanation: During user authentication, an internal error occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1514 **Bugs in auth-options.c option processing.**

Explanation: sshd encountered an error while parsing authorization options in the authorized_keys file.

System action: The program ends.

System programmer response: Notify the user of errors in their authorized keys file.

FOTS1529 **auth_rsa_verify_response: RSA modulus too small:** *bits* **< minimum** *minbits* **bits**

Explanation: During RSA authentication, the number of bits *bits* in the key was found to be too small. It needs to be bigger than *minbits*.

System action: The program continues.

System programmer response: Notify the user their key is too small.

FOTS1530 **auth_rsa_generate_challenge: BN_new() failed**

Explanation: During RSA authentication in sshd, a call to the OpenSSL function BN_new() failed. An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1531 **auth_rsa_generate_challenge: BN_CTX_new failed**

Explanation: During RSA authentication in sshd, a call to the OpenSSL function BN_CTX_new() failed. An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1532 **auth_rsa_verify_response: bad challenge length** *length*

Explanation: During RSA authentication in sshd, the challenge length was found to be too short. An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1533 **auth_rsa_challenge_dialog: BN_new() failed**

Explanation: During RSA authentication in sshd, a call to the OpenSSL function BN_new() failed. An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1555 **__tcsetcp() failed:** *system error*

Explanation: A call to __tcsetcp() failed while sshd was trying to set the code set for the master pty. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1556 *ttyname***:** *system error*

Explanation: A call to open() failed for *ttyname*. The system error is displayed with this message.

System action: The program ends if a pty is required.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1557 **chown** *ttyname* **0 0 failed:** *system error*

Explanation: A call to chown() failed while sshd was trying to release the pty and return ownership to uid 0. The system error is displayed with this message.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1558 **chmod** *ttyname* **0666 failed:** *system error*

Explanation: A call to chmod() failed while sshd was trying to release the pty and make the permissions 666.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1562 **setsid:** *system error*

Explanation: A call to setsid() failed while sshd was trying to make the tty the process controlling tty. The system error is displayed with this message.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1563 **Failed to disconnect from controlling tty.**

Explanation: A call to open() failed while sshd was tried to open the controlling tty with O_RDWR and O_NOCTTY. The system error is displayed with this message.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1567 **open /dev/tty failed – could not set controlling tty:** *system error*

Explanation: A call to open() failed for /dev/tty. The system error is displayed with this message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time*

Library Reference for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1568 **chown(***ttyname***, *userid*, *groupid*) failed:** *system error*

Explanation: sshd is attempting to change the owner and group of the tty *ttyname* to that of *userid* and *groupid* respectively. The call to chown() failed because the file system is read-only. The current owner of the tty is already that of *userid* or of a superuser.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1569 **chmod(***ttyname***, *mode*) failed:** *system error*

Explanation: sshd is attempting to change the permissions of the tty *ttyname* to that of *mode*. The call to chmod() failed because the file system is read-only. The current permissions allow read access for group and other.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1572 **stat(***ttyname***) failed:** *system error*

Explanation: A call to stat() failed for *ttyname*. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1573 **chown(***ttyname***, *userid*, *groupid*) failed:** *system error*

Explanation: sshd is attempting to change the owner and group of the tty *ttyname* to that of *userid* and *groupid* respectively. A call to chown() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1574 **chmod(***ttyname***, *mode*) failed:** *system error*

Explanation: sshd is attempting to change the permissions of the tty *ttyname* to that of *mode*. The call to chmod() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1575 **login_get_lastlog: Cannot find account for uid** *uid*

Explanation: A call to getpwuid() failed for UID *uid*.

System action: The program ends.

System programmer response: Verify there is a user account for *uid*. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1576 **login_init_entry: Cannot find user "***userid***"**

Explanation: sshd was unable to find the definition for user id *userid*. A call to getpwuid() failed.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1577 **This platform does not support both privilege separation and compression**

Explanation: The configuration options Compression and UsePrivilegeSeparation were both enabled. IBM z/OS does not support both privilege separation and compression.

System action: Compression is disabled and the program continues.

System programmer response: Determine if compression is necessary for your network.

FOTS1578 **Compression disabled**

Explanation: The configuration options Compression and UsePrivilegeSeparation were both enabled. IBM z/OS does not support both privilege separation and compression, so compression is disabled.

System action: The program continues.

System programmer response: Determine if compression is necessary for your network.

FOTS1579 *filename*: **line** *line number*: **Bad configuration option:** *configuration option*

Explanation: An option specified in an sshd configuration file is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the invalid option.

FOTS1581 **bad addr or host:** *address* **(***system error***)**

Explanation: The sshd daemon failed when trying to get the address information for *address*. The system error is displayed with this message.

System action: The program ends.

User response: Verify *address* is valid.

FOTS1582 *filename* **line** *line number*: **ports must be specified before ListenAddress.**

Explanation: In the sshd configuration file, the Port option was not specified before the ListenAddress option.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the option which caused this error. Change the order of these options in the sshd configuration file and reissue **sshd**.

FOTS1583 *filename* **line** *line number*: **too many ports.**

Explanation: The sshd Port option was specified more times than sshd supports. The maximum number of ports allowed by sshd is 256.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Port option which caused this error. Reissue sshd with a valid number of ports.

System action: The program ends.

FOTS1584 *filename* **line** *line number*: **missing port number.**

Explanation: The sshd configuration file *filename* has the Port option, but is missing the corresponding port number.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Port option, add a port number, and reissue sshd.

FOTS1585 *filename* **line** *line number*: **Badly formatted port number.**

Explanation: The sshd configuration file *filename* has the Port option, but the corresponding port number has caused a syntax error.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Port option, correct the port number, and reissue sshd.

FOTS1586 *filename* **line** *line number*: **missing integer value.**

Explanation: The sshd configuration file *filename* has a configuration option which expects an integer argument, but the argument is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the failing configuration option, add an integer argument, and reissue **sshd**.

FOTS1587 *filename* **line** *line number*: **missing time value.**

Explanation: The sshd configuration file *filename* has a configuration option which expects a time value, but the corresponding time value is missing. Options which expect time values include LoginGraceTime, KeyRegenerationInterval, and ClientAliveInterval.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the failing option, add a time value and reissue sshd.

FOTS1588 *filename* **line** *line number*: **invalid time value.**

Explanation: The sshd configuration file *filename* has a configuration option which expects a time value, but the corresponding time value is invalid. Options which expect time values include LoginGraceTime, KeyRegenerationInterval, and ClientAliveInterval.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the failing option, correct the time value and reissue **sshd**.

FOTS1589 *filename* **line** *line number*: **missing address**

Explanation: The sshd configuration file *filename* has the ListenAddress option, but the corresponding internet address on which to listen is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the ListenAddress option, add an internet address, and reissue **sshd**.

FOTS1590 *filename* **line** *line number*: **bad ipv6 inet addr usage.**

Explanation: The sshd configuration file *filename* has the ListenAddress option. The corresponding ipv6 internet address on which to listen is the wrong syntax. A left-bracket is missing a corresponding right bracket.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the ListenAddress option, correct the internet address, and reissue **sshd**.

FOTS1591 *filename* **line** *line number*: **bad address:port usage.**

Explanation: The sshd configuration file *filename* has the ListenAddress option. The corresponding internet address on which to listen is the wrong syntax. A port number should follow the colon.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the ListenAddress option, correct the internet address, and reissue **sshd**.

FOTS1592 *filename* **line** *line number*: **bad port number.**

Explanation: The port number specified with sshd configuration option ListenAddress is invalid. It should be a number greater than 0 and less than or equal to 65535.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the ListenAddress option, correct the port specification, and reissue **sshd**.

FOTS1593 *filename* **line** *line number*: **bad inet addr usage.**

Explanation: The sshd configuration file *filename* has the ListenAddress option. The corresponding internet address or host on which to listen is the wrong syntax. Invalid data appears where a port specification might be.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the ListenAddress option, correct the port specification, and reissue **sshd**.

FOTS1594 *filename* **line** *line number*: **too many host keys specified (max** *hostkeys*).

Explanation: The maximum number of host keys and host key ring certificates that can be specified in configuration files or the command line has been exceeded.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the HostKey or HostKeyRingLabel keywords which caused this error. Reissue **sshd** with a valid number of HostKey or HostKeyRingLabel keywords.

FOTS1595 *filename* **line** *line number*: **missing file name.**

Explanation: The sshd configuration file *filename* has a configuration option specified which expects a filename argument. The filename argument for this option is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the configuration option which caused this error, and add a filename. Reissue **sshd**.

FOTS1596 *filename* **line** *line number*: **missing yes/without–password/ forced–commands–only/no argument.**

Explanation: The sshd configuration file *filename* has the PermitRootLogin option specified. The argument for this option is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the PermitRootLogin option which caused this error, and add an argument. Reissue **sshd**.

FOTS1597 *filename* **line** *line number*: **Bad yes/without–password/ forced–commands–only/no argument:** *arg*

Explanation: The sshd configuration file *filename* has the PermitRootLogin option specified. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the PermitRootLogin option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1598 *filename* **line** *line number*: **missing yes/no argument.**

Explanation: The sshd configuration file *filename* has a configuration option specified which expects a yes/no argument. The argument for this option is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the configuration option which caused this error, and add an argument. Reissue **sshd**.

FOTS1599 *filename* **line** *line number*: **Bad yes/no argument:** *arg*

Explanation: The sshd configuration file *filename* has a configuration option specified which expects a yes/no argument. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the configuration option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1601 *filename* **line** *line number*: **unsupported log facility '***arg***'**

Explanation: The sshd configuration file *filename* has the SyslogFacility option specified. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the SyslogFacility option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1602 *filename* **line** *line number*: **unsupported log level '***arg***'**

Explanation: The sshd configuration file *filename* has the LogLevel option specified. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the LogLevel option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1603 *filename* **line** *line number*: **too many allow users.**

Explanation: The sshd AllowUsers option was specified more times than sshd supports. The maximum number of AllowUsers specifications allowed by sshd is 256.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the AllowUsers option which caused this error. Reissue **sshd** with a valid number of AllowUsers options.

FOTS1604 *filename* **line** *line number*: **too many deny users.**

Explanation: The sshd DenyUsers option was specified more times than sshd supports. The maximum number of DenyUsers specifications allowed by sshd is 256.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the DenyUsers option which caused this error. Reissue sshd with a valid number of DenyUsers options

---

FOTS1605 *filename* **line** *line number*: **too many allow groups.**

Explanation: The sshd AllowGroups option was specified more times than sshd supports. The maximum number of AllowGroups specifications allowed by sshd is 256.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the AllowGroups option which caused this error. Reissue **sshd** with a valid number of AllowGroups options.

FOTS1606 *filename* **line** *line number*: **too many deny groups.**

Explanation: The sshd DenyGroups option was specified more times than sshd supports. The maximum number of DenyGroups specifications allowed by sshd is 256.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the DenyGroups option which caused this error. Reissue **sshd** with a valid number of DenyGroups options.

FOTS1607 *filename* **line** *line number*: **Missing argument.**

Explanation: The sshd configuration file *filename* has the Ciphers, MACs, or Protocol option specified. The argument for this option is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the option which caused this error, and add an argument. Reissue **sshd**.

FOTS1608 *filename* **line** *line number*: **Bad SSH2 cipher spec '***arg***'.**

Explanation: The sshd configuration file *filename* has the Ciphers option specified. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Ciphers option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1610 *filename* **line** *line number*: **Bad SSH2 mac spec '***arg***'.**

Explanation: The sshd configuration file *filename* has the MACs option specified. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the MACs option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1611 *filename* **:** *message*

Explanation: A call to fopen() failed on file *filename*. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1612 *filename* **line** *line number*: **Bad protocol spec '***arg***'.**

Explanation: The sshd configuration file *filename* has the Protocol option specified. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Protocol option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1613 *filename* **line** *line number*: **too many subsystems defined.**

Explanation: The sshd Subsystem option was specified more times than sshd supports. The maximum number of Subsystem specifications allowed by sshd is 256.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Subsystem option which caused this error. Reissue sshd with a valid number of Subsystem options.

FOTS1614 *filename* **line** *line number*: **Missing subsystem name.**

Explanation: The sshd configuration file *filename* has the Subsystem option specified. The argument for this option is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the option which caused this error, and add an argument. Reissue **sshd**.

FOTS1615 *filename* **line** *line number*: **Subsystem '***name***' already defined.**

Explanation: The sshd configuration file *filename* has the Subsystem option specified. The subsystem *name* is already defined.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Subsystem option which caused the error.

FOTS1616 *filename* **line** *line number*: **Missing subsystem command.**

Explanation: The sshd configuration file *filename* has the Subsystem option specified. The command argument for this option is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the Subsystem option which caused the error.

FOTS1617 *filename* **line** *line number*: **Missing MaxStartups spec.**

Explanation: The sshd configuration file *filename* has the MaxStartups option specified. The argument for this option is missing.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the option which caused this error, and add an argument. Reissue **sshd**.

FOTS1618 *filename* **line** *line number*: **Illegal MaxStartups spec.**

Explanation: The sshd configuration file *filename* has the MaxStartups option specified. The argument *arg* for this option is invalid.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the MaxStartups option which caused this error, and correct the argument. Reissue **sshd**.

FOTS1619 **server_input_global_request: no/invalid user**

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1620 *filename* **line** *line number***: Missing handler for opcode** *arg* **(***opcode***)**

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1621 *filename* **line** *line number***: garbage at end of line; "***arg***".**

Explanation: The sshd configuration file *filename* contains the invalid data *arg*.

System action: The program ends.

System programmer response: Check *line number* of the sshd configuration file *filename* for the data which caused this error, and correct the argument. Reissue **sshd**.

FOTS1622 *filename***: terminating,** *options* **bad configuration options**

Explanation: sshd encountered too many bad configuration options in *filename*.

System action: The program ends.

System programmer response: Check the sshd configuration file *filename* for the data which caused this error, and correct the argument. Reissue **sshd**.

FOTS1623 **pipe(notify_pipe) failed** *system error*

Explanation: A call to pipe() failed. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1624 **fcntl(notify_pipe, F_SETFD) failed** *system error*

Explanation: A call to fcntl() failed. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1625 **select:** *system error*

Explanation: A call to select() failed. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1626 **Strange, wait returned pid** *pid1***, expected** *pid2*

Explanation: A call to waitpid() returned *pid1* but sshd expected *pid2*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1627 **server_input_global_request: no user**

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1628 **authentication forwarding requested twice.**

Explanation: The remote ssh client has requested agent forwarding twice.

System action: The program continues.

System programmer response: Follow local procedures for handling multiple agent forwarding requests.

FOTS1629 **setsid failed:** *system error*

Explanation: A call to setsid() failed while sshd was trying to create a new session and process group. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1630 dup2 stdin: *system error*

Explanation: A call to dup2() failed for stdin. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1631 dup2 stdout: *system error*

Explanation: A call to dup2() failed for stdout. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1632 dup2 stderr: *system error*

Explanation: A call to dup2() failed for stderr. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1633 passwd

Explanation: A attempt to exec the passwd utility failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1634 setlogin failed: *system error*

Explanation: A call to setlogin() failed. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1635 no more sessions

Explanation: Too many session channels were attempted to be opened in sshd. The maximum number of session channels allowed by sshd is 10.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1636 session_by_pid: unknown pid *pid*

Explanation: ssh attempted to get a session id from the pid number *pid*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1637 session_pty_req: session *sessionid* **alloc failed**

Explanation: While sshd was requesting a pty for the session *sessionid*, a pty could not be allocated.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1638 subsystem: cannot stat *command***:** *system error*

Explanation: While sshd was attempting to run a subsystem, the command for the subsystem failed. Specifically, a call to stat() failed for the command. The system error is displayed with this message.

System action: The program continues.

System programmer response: Verify that the command specified for the subsystem (in the sshd configuration file) is in the search order specified by PATH. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1639 session_pty_cleanup: no session

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1640 close(s->ptymaster/*ptynum*): *system error*

Explanation: While sshd was attempting to close the pty, a call to close() failed. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1641 no user for session *sessionid*

Explanation: sshd cannot find a user associated with session *sessionid*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1642 Can't get IP address for X11 DISPLAY.

Explanation: While ssh was attempting to set up X11 forwarding, a call to gethostbyname() failed.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1643 dup2 stdin

Explanation: A call to dup2() failed for stdin. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer. Follow local procedures for reporting problems to IBM.

FOTS1644 dup2 stdout

Explanation: A call to dup2() failed for stdout. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

FOTS1645 dup2 stderr

Explanation: A call to dup2() failed for stderr. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact

FOTS1646 *shell_program* : *message*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1647 *shell_program* : *message*

Explanation: A call to execve() failed on executing *shell_program*. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1650 setgid

Explanation: A call to setgid() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact

FOTS1651 initgroups

Explanation: A call to initgroups() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact

FOTS1652 login

Explanation: An error occurred while sshd tried to execute the login program. A call to execl() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1657 do_exec_no_pty: no session

Explanation: An internal error occurred while sshd was attempting to execute a command with no tty.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1658 do_exec_pty: no session

Explanation: An internal error occurred while sshd was attempting to execute a command with a tty.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1659 child_set_env: too many env vars, skipping: *varname*

Explanation: sshd could not set the environment variable *varname* because the maximum allowed (1000) to be set has be reached.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1660 Too many lines in environment file *filename*

Explanation: sshd failed while reading the user's environment file because the file has exceeded the maximum number of lines (1000) supported by sshd.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1661 Failed to set uids to *uid***.**

Explanation: sshd failed to set the uid of the process to *uid*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1662 no user for session *sessionid*

Explanation: sshd could not find a user id associated with the session *sessionid*. An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1663 child_set_env: too many env vars

Explanation: sshd could not set an environment variable because the maximum allowed (1000) to be set has been reached.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1664 session_set_fds: called for proto != 2.0

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1665 no channel for session *sessionid*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1666 session_exit_message: session *sessionid***: no channel** *channel*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1667 gethostname: *system error*

Explanation: A call to gethostname() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1668 WARNING: Your password has expired.

Explanation: Your password has expired. You will be prompted to change it.

System action: The program ends.

User response: Enter your new password, and login again.

FOTS1669 Password change required but no TTY available.

Explanation: Your password has expired, but your session does not have a tty available from which to read the password.

System action: The program ends.

User response: Run a ssh session with a tty allocated, then change your password.

FOTS1671 Bad line *line number* **in** *filename*

Explanation: sshd failed while reading the user's environment file because it encountered a line with an invalid syntax.

System action: The program continues.

System programmer response: Notify the user their environment file has a syntax error on line *line number*.

FOTS1675 Could not run *filename*

Explanation: While sshd was running the user's startup files, a call to popen() failed while attempting to run *filename*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1679 Could not run *command*

Explanation: While sshd was running the user's startup files, a call to popen() failed while attempting to run *command*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1681 Could not chdir to home directory *dir***:** *system error*

Explanation: A call to chdir() failed while sshd was attempting to change to the user's home directory *dir*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1687 mm_make_entry(*address***): double address** *pointer***->***address2***(***size***)**

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1688 mmap(*size***):** *system error*

Explanation: While sshd was attempting to create a shared memory space, a call to mmap() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1689 munmap(*address***,** *size***):** *system error*

Explanation: While sshd was attempting to create a shared memory space, a call to munmap() failed. The system error is displayed with this message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS1690 mm_memvalid: address too large: *address*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1691 *function***: mm_malloc(***size***)**

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1692 mm_malloc: try to allocate 0 space

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1693 mm_malloc: size too big

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1694 **mm_free(***address1***): can not find** *address2*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1695 **mm_free(***address1***): double address** *address2*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1696 **mm_free: memory corruption:** *addr1***(***size***) >** *addr2*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1697 **mm_free: memory corruption:** *addr1* **<** *addr2***(***size***)**

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1698 **mm_memvalid: address too small:** *address*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1699 **mm_memvalid: end < address:** *address1* **<** *address2*

Explanation: An internal error has occurred.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1702 *function***: fd0** *file_descriptor* **!= 0**

Explanation: open() system call on /dev/null did not return 0.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1703 *function***: unexpected authentication from** *reqtype*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1704 *function***: authenticated invalid user**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1705 *function***: unpermitted request** *type*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1706 *function***: unsupported request:** *type*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1707 *function***: bad parameters:** *min want max*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1708 *function*: **data length incorrect:** *data_len*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1709 *function*: **no hostkey from index** *keyid*

Explanation: Internal error

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1710 *function*: **key_sign failed**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1711 *function*: **multiple attempts for getpwnam**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1712 *function*: **no bsd auth session**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1713 *function*: **key type and protocol mismatch**

Explanation: Key type does not match protocol being used.

System action: The program ends.

User response: Verify key is correct type. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1714 *function*: **unknown key type** *type*

Explanation: Unknown key type.

System action: The program ends.

User response: Verify key type. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1715 *function*: **bad key, not previously allowed**

Explanation: Bad key.

System action: The program ends.

User response: Verify key is correct. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1716 *function*: **bad public key blob**

Explanation: Public key data is bad.

System action: The program ends.

User response: Verify public key file is correct. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1717 *function*: **bad signature data blob**

Explanation: Key signature data is bad.

System action: The program ends.

User response: Verify key file is correct. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1718 *function*: **dup2**

Explanation: dup2() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local

procedures for reporting problems to IBM.

FOTS1719 *function*: **open(/dev/null):** *error_message*

Explanation: open() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1720 *function*: **BN_new**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1721 *function*: **bad ssh1 session id**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1723 *function*: **key_to_blob failed**

Explanation: Key error.

System action: The program ends.

User response: Verify key file is correct. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1724 *function*: **authctxt not valid**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1725 *function*: **bad key, not previously allowed**

Explanation: Key error.

System action: The program ends.

User response: Verify key file is correct. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1726 *function*: **key type mismatch**

Explanation: Key error.

System action: The program ends.

User response: Verify key file is correct. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1727 *function*: **received bad key**

Explanation: Key error.

System action: The program ends.

User response: Verify key file is correct. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1729 *function*: **no ssh1_challenge**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1730 **ssh–keysign not enabled in** *filename*

Explanation: EnableSSHKeysign is not enabled in the ssh configuration file *filename*.

System action: The program ends.

User response: Change the ssh configuration file to enable EnableSSHKeysign.

FOTS1731 **ssh_msg_send failed**

Explanation: A read or write failed during ssh-keysign processing.

System action: The program ends.

User response: Follow local procedures for reporting problems to IBM.

FOTS1733 *function*: **received bad response to challenge**

Explanation: Communication error.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1734 *function*: **auth too large**

Explanation: Communication error.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1735 **mm_get_get: internal error: bad session id**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1736 *function*: **bad request size**

Explanation: Communication error.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1738 *function*: **mm_zalloc(***ncount***, *size*)**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1739 **fcntl(***file_descriptor***, F_SETFD)**

Explanation: The fcntl() system call failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1740 *function*: **socketpair**

Explanation: socketpair() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1742 *filename*: **skipping, filename contains a newline**

Explanation: Filename contains a newline character.

System action: The command continues.

User response: Verify that the filename specified is correct.

FOTS1743 **pipe:** *error_message*

Explanation: pipe() system call failed.

System action: The command ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1744 *filename*: *error_message*

Explanation: A file operation failed on the specified file.

System action: The command continues.

User response: Verify that the file exists and has proper access permissions. If error persists contact your system programmer.

System programmer response: If specified file does not appear to have any problems, follow local procedures for reporting the problem to IBM.

FOTS1745 **unknown user** *userid*

Explanation: getpwuid() system call failed to return a user.

System action: The command ends.

User response: Verify that the specify user exists.

FOTS1748 *pathname*: **not a regular file**

Explanation: File specified is not a regular file.

System action: The command continues.

User response: Only specify regular files.

FOTS1750 *name/filename*: **name too long**

Explanation: Filename is too long.

System action: The command continues.

User response: Specify a filename less than 1100 characters long.

FOTS1753 **ambiguous target**

Explanation: Target specified on the command line is ambiguous.

System action: The command ends.

User response: Specify a nonambiguous target.

FOTS1754 *message*

Explanation: Connection error.

System action: The program ends.

User response: Verify connection and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1755 *user name*: **invalid user name**

Explanation: Invalid user name specified.

System action: The program continues.

User response: Specify a valid username.

FOTS1756 **RSA_blinding_on failed**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1757 **Hostbased authentication not enabled in** *config_file*

Explanation: The user attempted Hostbased authentication, but it is not enabled.

System action: The program ends.

User response: Enable host based authentication in configuration file.

FOTS1758 **could not open any host key**

Explanation: Could not open any host keys.

System action: The program ends.

User response: Verify that host keys exist, and that access permissions are properly set.

FOTS1759 **getpwuid failed**

Explanation: getpwuid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1760 **no hostkey found**

Explanation: No host key found.

System action: The program ends.

User response: Verify that host keys exist, and that access permissions are properly set.

FOTS1761 **ssh_msg_recv failed**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1762 **bad version**

Explanation: SSH version is not correct.

System action: The program end.

User response: Verify that you are running the proper version of SSH.

FOTS1763 bad fd

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1764 cannot get sockname for fd

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1765 not a valid request

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1766 no matching hostkey found

Explanation: No matching host key found.

System action: The program ends.

User response: Verify that the host keys exist, and access permissions are properly set.

FOTS1767 key_sign failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1768 *pathname***: set times:** *error_message*

Explanation: utimes() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1770 *program* **:** *message*

Explanation: A call to execvp() failed. The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1771 *path***: truncate:** *error_messages*

Explanation: ftruncate() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1772 *path***: set mode:** *error_message*

Explanation: chmod() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1776 protocol error: *error_message*

Explanation: scp error.

System action: The program ends.

User response: This is a catchall for a number of scp errors. See the error message at the end of this message for the specific error that occurred.

FOTS1778 fstat: *error_message*

Explanation: fstat() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1779 unexpected <newline>

Explanation: Unexpected newline in buffer read from socket.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1780 lost connection

Explanation: Connection Lost.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1781 mtime.sec not delimited

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1782 mtime.usec not delimited

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1783 atime.sec not delimited

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1784 atime.usec not delimited

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1785 expected control record

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1786 bad mode

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1787 mode not delimited

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1788 size not delimited

Explanation: Buffer read from socket is not in proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local

procedures for reporting problems to IBM.

FOTS1789 setenv failed for _BPXK_SUID_FORK: *error_message*

Explanation: The setenv system call failed and sshd could not set _BPXK_SUID_FORK. This may cause the user's session to have incorrect properties, including jobname, region size, and SMF accounting information.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1790 error: unexpected filename: *filename*

Explanation: The buffer read from socket is not in the proper format.

System action: The program ends.

User response: Verify connectivity and remote host status. If the problem persists contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1791 received directory without –r

Explanation: The buffer read from socket did not have the expected –r recursive option.

System action: The program ends.

User response: Verify connectivity and remote host status. If the problem persists contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1801 Couldn't create socket: *error_message*

Explanation: socket() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1802 Couldn't connect to PRNGD port *tcp_port***:** *error_message*

Explanation: connect() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1803 Couldn't connect to PRNGD socket "*path***":** *error_message*

Explanation: connect() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1804 Couldn't write to PRNGD socket: *error_message*

Explanation: write() system call inside atomicio() failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1805 Couldn't read from PRNGD socket: *error_message*

Explanation: read() system call inside atomicio() failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1806 Couldn't wait for child '*cmd_string***' completion:** *error_message*

Explanation: waitpid() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1807 **bad entropy command,** *cmd_filename* **line** *line*

Explanation: Error in ssh_prng_cmds file.

System action: The program continues.

User response: Make sure the ssh_prng_cmds file is set up properly. See the **ssh–rand–helper** man page for information.

FOTS1808 **missing or bad command string,** *cmd_filename* **line** *linenum* **-- ignored**

Explanation: Error in ssh_prng_cmds file.

System action: The program continues.

User response: Make sure the ssh_prng_cmds file is set up properly. See the **ssh–rand–helper** man page for information.

FOTS1809 **missing command path,** *cmd_filename* **line** *linenum* **-- ignored**

Explanation: Error in ssh_prng_cmds file.

System action: The program continues.

User response: Make sure the ssh_prng_cmds file is set up properly. See the **ssh–rand–helper** man page for information.

FOTS1810 **missing entropy estimate,** *cmd_filename* **line** *linenum* **-- ignored**

Explanation: Error in ssh_prng_cmds file.

System action: The program continues.

User response: Make sure the ssh_prng_cmds file is set up properly. See the **ssh–rand–helper** man page for information.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1811 **garbage at end of line** *linenum* **in** *cmd_filename*

Explanation: Error in ssh_prng_cmds file.

System action: The program continues.

User response: Make sure the ssh_prng_cmds file is set up properly. See the **ssh–rand–helper** man page for information.

FOTS1812 **ignored extra commands (max** *maximum***),** *filename* **line** *linenum*

Explanation: Error in ssh_prng_cmds file *filename*. The maximum number of command-line arguments passed to a command in the ssh_prng_cmds file has exceeded the internal limit of *maximum*.

System action: The program continues.

User response: Make sure the ssh_prng_cmds file is set up properly. See the **ssh–rand–helper** man page for information.

FOTS1813 **Invalid commandline option**

Explanation: Invalid command line option.

System action: The program continues.

User response: Enter a valid command line option.

FOTS1814 **You must specify a port or a socket**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1815 **Random pool path is too long**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1816 **Too many bytes to read from PRNGD**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1817 **Couldn't gettimeofday:** *error_message*

Explanation: gettimeofday() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1818 Couldn't open /dev/null: *error_message*

Explanation: open() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1819 Couldn't open pipe: *error_message*

Explanation: pipe() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1820 Couldn't fork: *error_message*

Explanation: fork() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1821 PRNG seedfile *filename* **is not a regular file**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1822 Couldn't get password entry for current user (*uid***):** *error_message*

Explanation: getpwuid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1823 problem writing PRNG seedfile *filename* **(***error_message***)**

Explanation: write() system call within atomicio() failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1824 PRNG seed extraction failed

Explanation: A call to the OpenSSL function RAND_bytes failed.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1825 could not open PRNG seedfile *filename* **(***error_message***)**

Explanation: open() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1826 couldn't read entropy commands file *cmdfilename***:** *error_message*

Explanation: fopen() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1827 Invalid number of output bytes

Explanation: Invalid number of bytes specified with -b option on the command line.

System action: The program ends.

User response: Specify a valid number of bytes. See man page for assistance.

FOTS1829 Entropy collection failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1830 PRNG initialisation failed -- exiting.

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1831 Not enough entropy in RNG

Explanation: Internal error.

System action: The program ends.

User response: Try reissuing the command. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1838 Couldn't fork: *error_message* **reason code =** *reasoncode*

Explanation: fork() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on reason code.

FOTS1840 mkdir *dirname***:** *error_message*

Explanation: The directory *dirname* could not be created. The mkdir() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1841 PRNG seed filename too long

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1842 problem renaming PRNG seedfile from *filename1* **to** *filename2* **(***error_message***)**

Explanation: The seedfile *filename1* could not be renamed. The rename() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1843 Couldn't extract entropy from PRNG

Explanation: Internal error.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1901 channel *channel***: protocol error: rcvd_oclose for istate** *istate*

Explanation: Invalid input from channel.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1902 channel *channel***: chan_read_failed for istate** *istate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1903 **channel** *channel*: **chan_ibuf_empty for non empty buffer**

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1904 **channel** *channel*: **chan_ibuf_empty for istate** *istate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1905 **channel** *channel*: **protocol error: rcvd_ieof for ostate** *ostate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1906 **channel** *channel*: **chan_write_failed for ostate** *ostate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1907 **channel** *channel*: **chan_obuf_empty for non empty buffer**

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1908 **channel** *channel*: **internal error: obuf_empty for ostate** *ostate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1909 **channel** *channel*: **cannot send ieof for istate** *istate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1910 **channel** *channel*: **cannot send oclose for ostate** *ostate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1911 **channel** *channel*: **protocol error: close rcvd twice**

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1912 **rsa_public_encrypt: BN_bin2bn failed**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1913 **channel** *channel***: cannot send eof for istate** *istate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1914 **channel** *channel***: cannot send close for istate/ostate** *istate*/*ostate*

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1915 **channel** *channel***: already sent close**

Explanation: Channel error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1916 **channel** *channel***: chan_shutdown_read: shutdown() failed for fd***socket* **[i***istate* **o***ostate***]:** *error_code*

Explanation: Channel error

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1917 **chan_set_istate: bad state** *ostate* **->** *next_state*

Explanation: Channel error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1918 **chan_set_ostate: bad state** *ostate* **->** *next_state*

Explanation: Channel error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1919 **fcntl O_NONBLOCK:** *error_message*

Explanation: The fcntl() system call failed. The system error is displayed with the message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS1920 **rsa_private_decrypt: BN_bin2bn failed**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1921 **setsockopt IPTOS_LOWDELAY:** *error_code*

Explanation: setsockopt() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1922 **setsockopt IPTOS_THROUGHPUT:** *error_code*

Explanation: setsockopt() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1923 **packet_set_connection: cannot load cipher 'none'**

Explanation: Error loading ciphers.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1924 **packet_set_seqnr: bad mode** *mode*

Explanation: Packet error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1925 **Compression already enabled.**

Explanation: Program attempted to enable compression when it is already active.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1926 **packet_set_encryption_key: unknown cipher number** *number*

Explanation: Cipher error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1927 **packet_set_encryption_key: keylen too small:** *keylen*

Explanation: Key length is less than 20.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1928 **packet_set_encryption_key: keylen too big:** *keylen*

Explanation: Key length is greater than SSH_SESSION_KEY_LENGTH.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1929 **newkeys: no keys for mode** *mode*

Explanation: Packet error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1930 **Read from socket failed:** *error_code*

Explanation: read() function call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1931 **padding error: need** *size_needed* **block** *block_size* **mod** *modulus*

Explanation: The needed size is not a multiple of the block size.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1932 **packet_disconnect called recursively.**

Explanation: Recursive invocation of packet_disconnect.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1933 Write failed: *error_code*

Explanation: write() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1934 put_host_port: asprintf: *error_message*

Explanation: The asprintf() call failed. The error is displayed with the message.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1935 addargs: argument too long

Explanation: The vasprintf() call failed. An argument was too long and could not be added to the argument string.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1936 replacearg: argument too long

Explanation: The vasprintf() call failed. An argument was too long and could not be replaced in the argument string.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1937 replacearg: tried to replace invalid arg *argument_number* **>=** *total_arguments*

Explanation: Argument *argument_number* does not identify a valid argument to replace.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1938 tilde_expand_filename: username too long

Explanation: Unable to complete tilde expansion for the specified filename. The user name is too long.

System action: The program ends.

User response: Verify that the user name is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1939 tilde_expand_filename: No such user *user_name*

Explanation: Unable to complete tilde expansion for the specified filename. The user name *user_name* is not valid.

System action: The program ends.

User response: Verify that the user name is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1940 tilde_expand_filename: No such uid *UID*

Explanation: Unable to complete tilde expansion for the specified filename. The UID *UID* is not valid.

System action: The program ends.

User response: Verify that the UID is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1941 Couldn't open /dev/null: *error_message*

Explanation: The open() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1942 tilde_expand_filename: Path too long

Explanation: The expanded filename is too long.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local

| procedures for reporting problems to IBM.

FOTS1943 rsa_generate_additional_parameters: BN_sub/mod failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1944 Couldn't read from ssh-rand-helper: *error_message*

Explanation: read() system call failure from **ssh–rand–helper**.

System action: The program ends.

User response: Verify all ssh components are installed and configured correctly. Refer to the *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Verify all ssh components are installed and configured correctly. If error persists follow local procedures for reporting problems to IBM.

FOTS1945 ssh-rand-helper child produced insufficient data

Explanation: Error with pseudo-random number generating functions.

System action: The program ends.

User response: This error often occurs due to errors in installation and setup of ssh. Verify all ssh components are installed and configured correctly. If error persists contact your system programmer to report the error.

System programmer response: Verify all ssh components are installed and configured correctly. If error persists follow local procedures for reporting problems to IBM.

FOTS1946 Couldn't wait for ssh–rand–helper completion: *error_message*

Explanation: waitpid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1947 ssh-rand-helper terminated abnormally

Explanation: Error with pseudo-random number generating functions.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1948 ssh-rand-helper exit with exit status *exit_status*

Explanation: Error with pseudo-random number generating functions.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1949 PRNG is not seeded

Explanation: OpenSSL error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1950 OpenSSL version mismatch. Built against *req_version***, you have** *cur_version*

Explanation: OpenSSL error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1951 getuid: *error_message*

Explanation: getuid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1952 **geteuid:** *error_message*

Explanation: geteuid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1953 **(rand child) setuid(***orig_uid***):** *error_message*

Explanation: setuid() or seteuid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1954 **(rand child) Couldn't exec '***path***':** *error_message*

Explanation: execl() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1955 **ssh_askpass: fflush:** *error_message*

Explanation: fflush() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1956 **ssh_askpass: pipe:** *error_message*

Explanation: pipe() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1957 **ssh_askpass: fork:** *error_message*

Explanation: fork() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1958 **internal error: askpass undefined**

Explanation: Internal error

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1959 **ssh_askpass: dup2:** *error_message*

Explanation: dup2() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1960 **ssh_askpass: exec(***path***):** *error_message*

Explanation: execlp() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1961 **rsa_private_decrypt() failed**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1962 rsa_public_encrypt() exponent too small or not odd

Explanation: RSA exponent value is bad.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1963 rsa_public_encrypt() failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1964 rsa_generate_additional_parameters: BN_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS1965 rsa_generate_additional_parameters: BN_CTX_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2003 ssh_dss_sign: no DSA key

Explanation: DSA key not found or wrong type.

System action: The program continues.

User response: Verify DSA key. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2004 ssh_dss_sign: sign failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2005 bad sig size *rlen slen*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2006 ssh_dss_verify: no DSA key

Explanation: DSA key not found or wrong type.

System action: The program continues.

User response: Verify DSA key. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2007 ssh_dss_verify: cannot handle type *ktype*

Explanation: DSA key type error.

System action: The program continues.

User response: Verify DSA key. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2008 ssh_dss_verify: remaining bytes in signature *rlen*

Explanation: DSA key signature error.

System action: The program continues.

User response: Verify DSA key. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2009 bad sigbloblen *len* **!= SIGBLOB_LEN**

Explanation: Key signature error.

System action: The program ends.

User response: Verify DSA key. If error persists

contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2010 ssh_dss_verify: DSA_SIG_new failed

Explanation: Error generating DSA signature.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2011 ssh_dss_verify: BN_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2012 ssh_dss_verify: BN_bin2bn failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2013 ssh_rsa_sign: no RSA key

Explanation: RSA key not found or wrong type.

System action: The program continues.

User response: Verify RSA key exists and is correct type. If problem persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2014 ssh_rsa_sign: EVP_get_digestbynid *nid* **failed**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2015 ssh_rsa_sign: RSA_sign failed: *error_message*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2016 ssh_rsa_sign: slen *len1* **slen2** *len2*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2017 ssh_rsa_verify: no RSA key

Explanation: RSA key not found or wrong type.

System action: The program continues.

User response: Verify RSA key exists and is the correct type. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2018 ssh_rsa_verify: RSA modulus too small: *key_modulus* **< minimum** *rsa_min_modulus* **bits**

Explanation: Modulus for RSA key is too small.

System action: The program continues.

User response: Verify that the RSA key was properly generated. If the error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2019 ssh_rsa_verify: cannot handle type *key_type*

Explanation: The RSA key is not the proper type.

System action: The program continues.

User response: Verify RSA key exists and is the correct type. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2020 **ssh_rsa_verify: remaining bytes in signature** *rlen*

Explanation: RSA key signature error.

System action: The program continues.

User response: Verify RSA key exists and is the correct type. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2021 **ssh_rsa_verify: len** *len* **> modlen** *modlen*

Explanation: RSA key error.

System action: The program continues.

User response: Verify RSA key exists and is the correct type. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2022 **ssh_rsa_verify: EVP_get_digestbynid** *nid* **failed**

Explanation: RSA key error.

System action: The program continues.

User response: Verify RSA key exists and is the correct type. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2023 **bad hashlen**

Explanation: RSA key error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2024 **bad siglen**

Explanation: RSA key error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2025 **RSA_public_decrypt failed:** *error_string*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2026 **bad decrypted len:** *len* **!=** *hlen* **+** *oidlen*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2027 **oid mismatch**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2028 **hash mismatch**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2029 **User name after tilde too long.**

Explanation: User name is greater than 100 characters.

System action: The program ends.

User response: User name must be less than 100 characters.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2030 **Unknown user** *user***.**

Explanation: Unknown user.

System action: The program ends.

User response: Verify that the user exists on the system. If error persists contact your system

programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2031 Home directory too long (*len* > *maxpathlen*)

Explanation: The pathlen of the home directory exceeds *maxpathlen*.

System action: The program ends.

User response: Home directory cannot exceed *maxpathlen* characters.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2032 cfsetispeed failed for *baud*

Explanation: TTY error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2033 cfsetospeed failed for *baud*

Explanation: TTY error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2034 getgroups: *error_message*

Explanation: getgroups()system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2035 initgroups: *pw_name*: *error_message*

Explanation: initgroups() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2036 *function*: **was able to restore old [e]gid"**

Explanation: The function *function* failed because the process was able to switch back to its original group id. Internal error.

System action: The program ends.

User response: Follow local procedures for reporting problems to IBM.

FOTS2037 setgroups: *error_message*

Explanation: setgroups() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2038 setegid *gid*: *error_message*

Explanation: setegid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2039 seteuid *uid*: *error_message*

Explanation: seteuid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2040 restore_uid: temporarily_use_uid not effective

Explanation: Error restoring original uid.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2041 *function***: egid incorrect gid:***gid* **egid:***egid* **(should be** *newgid***)**

Explanation: The function *function* failed because the process was able to switch back to its original group id. Internal error. *gid* is the current group id of the process. *egid* is the current effective group id of the process. *newgid* is the group id the process should be running as.

System action: The program ends.

User response: Follow local procedures for reporting problems to IBM.

FOTS2042 *function***: was able to restore old [e]gid"**

Explanation: The function *function* failed because the process was able to switch back to its original user id. Internal error.

System action: The program ends.

User response: Follow local procedures for reporting problems to IBM.

FOTS2043 *function***: euid incorrect uid:***uid* **euid:***euid* **(should be** *newuid***)**

Explanation: The function *function* failed because the process was able to switch back to its original user id. Internal error. *uid* is the current user id of the process. *euid* is the current effective user id of the process. *newuid* is the user id the process should be running as.

System action: The program ends.

User response: Follow local procedures for reporting problems to IBM.

FOTS2044 **permanently_set_uid: temporarily_use_uid effective**

Explanation: Error setting uid.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2045 **setgid** *gid***:** *error_message*

Explanation: setgid() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2046 **setuid** *UID***:** *error_message*

Explanation: The setuid() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2047 **xmalloc: zero size**

Explanation: Call to xmalloc specified zero size.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2048 **xmalloc: out of memory (allocating** *size* **bytes)**

Explanation: Unable to allocate requested number of bytes.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2049 **xrealloc: zero size**

Explanation: Call to xrealloc specified zero size.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2050 **xrealloc: out of memory (new_size** *size* **bytes)**

Explanation: Unable to allocate requested number of bytes.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2051 xfree: NULL pointer given as argument

Explanation: NULL pointer given as argument to xfree.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2052 newkeys_from_blob: remaining bytes in blob *len*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2053 *function*: **newkey == NULL**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2054 close(s–>ptymaster): *error_message*

Explanation: close() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2055 *function*: **write**

Explanation: Failure writing to a socket.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2056 xcalloc: zero size

Explanation: The call to xcalloc() specified size of zero.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2057 *function*: **read:** *return_value*

Explanation: Could not read from a socket.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2058 *function*: **read: bad msg_len** *msg_len*

Explanation: Message read from socket is too long.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2059 *function*: **read:** *ret_value* **!= msg_len**

Explanation: Number of bytes read from socket is incorrect.

System action: The program ends.

User response: Verify connectivity and remote machine status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2060 *function*: **read: rtype** *rtype* **!= type** *type*

Explanation: Type read from socket does not match type expected.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2061 *function*: **MONITOR_ANS_MODULI failed**

Explanation: Response received is not correct.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2062 *function*: **BN_new failed**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2063 **xcalloc: nmemb * size > SIZE_T_MAX**

Explanation: The call to xcalloc() specified a size that is too large.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2064 *function*: **struct passwd size mismatch**

Explanation: passwd structure received is not the correct size.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2065 *function*: **bad ivlen: expected** *block_size* **!=** *len*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2066 *function*: **bad cipher name** *name* **or pointer** *cipher*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2067 *function*: **can not setup mac** *mac_name*

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2068 *function*: **bad mac key length:** *len* **>** *mac_len*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2069 *function*: **conversion of newkeys failed**

Explanation: Error converting keys.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2070 **key_from_blob: can't read key type**

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2071 *function*: **key_from_blob failed**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2072 key_from_blob: can't read rsa key

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2073 *function*: **key_to_blob failed**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2074 key_from_blob: can't read dsa key

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2075 *function*: **reply from monitor too large**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2076 *function*: **sendmsg(***fd***):** *error_message*

Explanation: sendmsg() system call failed.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2077 *function*: **sendmsg: expected sent 1 got** *len*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2078 *function*: **UsePrivilegeSeparation=yes not supported**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2079 *function*: **recvmsg:** *system error*

Explanation: recvmsg() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2080 *function*: **recvmsg: expected received 1 got** *len*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2082 *function*: **expected type** *SCM_RIGHTS* **got** *cmsg_type*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2083 percent_expand: NULL replacement

Explanation: Unable to expand escape characters. A NULL escape character was found.

System action: The program ends.

User response: Verify that the escape characters are valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2088 percent_expand: too many keys

Explanation: Unable to expand escape characters. Too many escape characters were specified.

System action: The program ends.

User response: Verify that the escape characters are valid and don't exceed the limit, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2089 percent_expand: string too long

Explanation: Unable to expand escape characters. The resulting string is too long.

System action: The program ends.

User response: Verify that the escape characters are valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2090 XXX too many packets with same key

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2091 setsockopt IP_TOS *tos***:** *message***:**

Explanation: setsockopt() system call failed.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2092 percent_expand: unknown key %*escape_key*

Explanation: Unable to expand escape character. An unknown escape character *%escape_character* was specified.

System action: The program ends.

User response: Verify that the escape characters are valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2093 xcalloc: out of memory (allocating *size* **bytes)**

Explanation: Unable to allocate the requested number of bytes *size*.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2094 xasprintf: could not allocate memory

Explanation: Unable to allocate the requested number of bytes.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2095 xrealloc: nmemb * size > SIZE_T_MAX

Explanation: The call to xrealloc() specified a size that is too large.

System action: The program ends.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2096 WARNING: *filename* **does not exist, using fixed modulus**

Explanation: The fopen() system call failed to open file *filename*. Fixed modulus will be used.

System action: The program continues.

User response: Verify that the file *filename* exists, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2097 WARNING: no suitable primes in *filename*

Explanation: No suitable primes were found in file *filename*. Fixed modulus will be used.

System action: The program continues.

User response: Verify that the contents of file *filename* are valid, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2098 Warning: *filename*, **line** *line_number*: **keysize mismatch for host** *host_name*: **actual** *actual_keysize* **vs. announced** *announced_keysize*.

Explanation: The keysize *announced_keysize* on line *line_number* in file *filename* is incorrect. The correct keysize is *actual_keysize*.

System action: The program continues.

User response: Correct the keysize, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2099 Warning: replace *announced_keysize* **with** *actual_keysize* **in** *filename*, **line** *line_number*.

Explanation: The keysize *announced_keysize* on line *line_number* in file *filename* is incorrect. The correct keysize is *actual_keysize*.

System action: The program continues.

User response: Correct the keysize, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2101 No key to look up!

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2102 Error calculating host key fingerprint.

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2103 dns_export_rr: unsupported algorithm

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2104 Too many bits: *bits* > *TEST_MAXIMUM*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2105 Too few bits: *bits* < *TEST_MINIMUM*

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2106 Insufficient memory for tiny sieve: need *bytes* **bytes**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2107 Insufficient memory for small sieve: need *bytes* **bytes**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2108 Error writing to modulus candidate file: *error_message*

Explanation: A call to fflush() failed on file *filename* The system error is displayed with this message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2109 BN_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2110 BN_copy: failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2111 BN_set_bit: failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2112 BN_set_word failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2113 BN_add failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2114 BN_CTX_new failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2115 BN_hex2bn failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2116 kexdh_client: BN_bin2bn failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2117 *function*: set_nonblock(*file_descriptor*)

Explanation: Unable to set file descriptor *file_descriptor* to non–blocking. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2118 channel_add_adm_permitted_opens: too many forwards

Explanation: Too many port forwarding destinations specified for the sshd_config PermitOpen keyword.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid sshd_config PermitOpen keyword values, and try the request again.

FOTS2119 channel_prepare_select: max_fd (*maximum_file_descriptor*) is too large

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2120 reverse mapping checking getaddrinfo for *host_name* [*ipaddr*] failed – POSSIBLE BREAK–IN ATTEMPT!

Explanation: When **sshd** attempted to map *host_name* back to an IP address, a call to getaddrinfo() failed. **sshd** will use the socket IP address rather than the returned hostname from the Domain Name System (DNS) server.

System action: The program continues.

System programmer response: Verify that the entries in the Domain Name System (DNS) database are correct.

FOTS2121 get_socket_address: getnameinfo *flag* failed: *system error*

Explanation: A call to getnameinfo() failed with system error *system error*. *flag* is the argument of getnameinfo().

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2122 get_sock_port: getnameinfo NI_NUMERICSERV failed: *system error*

Explanation: A call to getnameinfo() failed with system error *system error*.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of argument NI_NUMERICSERV. Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2123 BN_rand failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2124 buffer_consume_ret: trying to get more bytes than in buffer

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2125 buffer_get: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2126 buffer_put_bignum: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2127 buffer_get_bignum_ret: invalid length

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2128 buffer_get_bignum_ret: BN_bin2bn failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2129 buffer_get_bignum_ret: buffer_consume failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2130 buffer_get_bignum: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2131 buffer_put_bignum2: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2132 buffer_get_bignum2_ret: invalid bignum

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2133 buffer_get_bignum2_ret: negative numbers not supported

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2134 buffer_get_bignum2_ret: BN_bin2bn failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2135 buffer_get_bignum2: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2136 BN_lshift failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2137 BN_add_word failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2138 BN_rshift failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2139 ssh_msg_recv: read: header

Explanation: Internal error. Partial data was read into an internal buffer.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2140 ssh_msg_recv: read: *error_message*

Explanation: Internal error. Partial data was read into an internal buffer. The system error is displayed with the message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2142 buffer_get_int: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local

| procedures for reporting problems to IBM.

FOTS2143 buffer_get_string_ret: buffer_get failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2144 buffer_get_string: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2145 buffer_get_char_ret: buffer_get_ret failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2146 buffer_get_char: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2147 buffer_get_string_bin: buffer error

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2148 buffer_get_string_bin_ret: buffer_get_ret failed

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2149 buffer_put_cstring_bin: s == NULL

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2150 RESTART FAILED: av[0]='*arg0*', error: *system error*.

Explanation: A SIGHUP signal was sent to sshd, but sshd was unable to restart. A call to execv() with the argument *argv0* failed.

System action: The program ends.

System programmer response: Attempt to run *arg0* manually. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2151 Could not write ident string to *ipaddr*

Explanation: A write to the socket failed while sshd was trying to send the SSH protocol version identification string to the peer.

System action: The daemon handling the connection ends.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2152 Did not receive identification string from *ipaddr*

Explanation: sshd could not read the remote system's version identification.

System action: The daemon handling the connection ends.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2153 Bad protocol version identification '*versionstring*' from *ipaddr*

Explanation: The local SSH daemon discovered a version incompatibility. sshd discovered that the remote system's version of SSH is not compatible with this version of SSH. The remote system is *ipaddr*. The version of SSH on the remote system is *versionstring*.

System action: The program ends.

System programmer response: Upgrade the SSH client on the remote system. Verify that the version on the remote system works properly.

FOTS2154 probed from *remote_ip* with *version*. Don't panic.

Explanation: During version identification exchange, sshd discovered that the remote system's version of SSH indicates it is a probe. The remote system is *remote_ip*. The version string of SSH that attempted a connection is *version*.

System action: The daemon handling the connection ends.

System programmer response: Follow local procedures for handling probes.

FOTS2155 scanned from *remote_ip* with *version*. Don't panic.

Explanation: During version identification exchange, sshd discovered that the remote system's version of SSH indicates it is a scanner, such as what might be sent by a ScanSSH program. The remote system is *remote_ip*. The version string of SSH that attempted a connection is *version*.

System action: The daemon handling the connection ends.

System programmer response: Follow local procedures for handling SSH scans.

FOTS2156 Protocol major versions differ for *remoteip*: *sversion* vs. *cversion*

Explanation: During version identification exchange, sshd discovered that the remote system's version of SSH, *cversion*, is not compatible with the local version of SSH, *sversion*. The remote system is *remote_ip*.

System action: The daemon handling the connection ends.

System programmer response: Verify that the remote version of SSH is compatible with the local version being run by the daemon. If compatible, follow local procedures for reporting problems to IBM.

FOTS2157 sshd: no hostkeys available -- exiting.

Explanation: During initialization, sshd could not find any host keys for either Protocol Version 1 or Protocol Version 2.

System action: The program ends.

System programmer response: Generate the host keys. See *IBM Ported Tools for z/OS User's Guide* for information on setting up the host keys for sshd.

FOTS2158 User *username* not allowed because shell *shell* does not exist

Explanation: sshd refused access to user *username* because the user's default program is set to *shell*, and *shell* does not exist.

System action: The program continues.

System programmer response: Follow local procedures for setting up user accounts.

FOTS2159 User *username* not allowed because shell *shell* is not executable

Explanation: sshd refused access to user *username* because the user's default program is set to *shell*, and *shell* is not marked as executable.

System action: The program continues.

System programmer response: If the intent is to allow access to the user, change the POSIX permissions of *shell* to make it executable. See the "chmod" command in *z/OS UNIX System Services Command Reference* for more information.

FOTS2160 User *username* not allowed because listed in DenyUsers

Explanation: sshd refused access to user *username* because the user was denied access through the DenyUsers keyword in the sshd_config file.

System action: The program continues.

System programmer response: None.

FOTS2161 User *username* not allowed because not listed in AllowUsers

Explanation: sshd refused access to user *username* because the username is not listed with the AllowUsers keyword in the sshd_config file.

System action: The program continues.

System programmer response: None.

FOTS2162 User *username* not allowed because not in any group

Explanation: sshd refused access to user *username* because the user does not have any groups associated with it.

System action: The program continues.

System programmer response: Follow local procedures for setting up user accounts.

FOTS2163 **User *username* not allowed because a group is listed in DenyGroups**

Explanation: sshd refused access to user *username* because the user belongs to a group which was denied access through the DenyGroups keyword in the sshd_config file.

System action: The program continues.

System programmer response: None.

FOTS2164 **User *username* not allowed because none of user's groups are listed in AllowGroups**

Explanation: sshd refused access to user *username* because the user belongs to a group which is not listed with the AllowGroups keyword in the sshd_config file.

System action: The program continues.

System programmer response: None.

FOTS2165 **ROOT LOGIN REFUSED FROM *ipaddr***

Explanation: sshd refused access to a superuser due to the setting of the PermitRootLogin keyword in the sshd_config file.

System action: The program continues.

System programmer response: None.

FOTS2166 **Authentication refused for *username*: bad owner or modes for *filename***

Explanation: sshd refused access to a user *username* because either the permissions on the user's hostfile *filename* are too open, the file is not owned by *username*, or a call to stat() failed for *filename*.

System action: The program continues.

System programmer response: Instruct the user to correct their setup.

FOTS2167 **User *username* from *ipaddr* not valid**

Explanation: sshd refused access to a user *username* because sshd does not recognize *username* as a valid user on the local system. Specifically, a call to getpwnam() for *username* failed.

System action: The program continues.

System programmer response: None.

FOTS2168 **Authentication tried for *username* with correct key but not from a permitted host (host=*hostname*, ip=*hostip*).**

Explanation: sshd refused access to a user *username* because the user's authorized_keys file has a "from=" option specification which does not permit *hostname* or *hostip*.

System action: The program continues.

System programmer response: None.

FOTS2169 **Bad options in *authfile* file, line *linenum*: *options***

Explanation: sshd refused access to a user because the user's authorized_keys file *authfile* has a bad options specification string *options* on line *linenum* of the file.

System action: The program continues.

System programmer response: None.

FOTS2170 **Client on *hostname* failed to respond correctly to host authentication."**

Explanation: sshd refused access to a user during RhostsRSAAuthentication because the ssh client on *hostname* did not respond correctly to the challenge.

System action: The program continues.

System programmer response: Check that the public host key for *hostname* is valid in the system-wide known hosts file. Instruct the user to verify that the public host key for *hostname* is valid in their known hosts file.

FOTS2171 **Rhosts authentication refused for *username*: no home directory *dirname***

Explanation: sshd refused access to user *username* because the user's HOME directory *dirname* does not exist or is inaccessible. A call to stat() for *dirname* failed.

System action: The program continues.

System programmer response: Follow local procedures for setting up user accounts.

FOTS2172 **Rhosts authentication refused for *username*: bad ownership or modes for home directory.**

Explanation: sshd refused access to user *username* because the user's HOME directory is writable by others, or is not owned by the user.

System action: The program continues.

System programmer response: Follow local procedures for setting up user accounts.

FOTS2173 **Rhosts authentication refused for *username*: bad modes for *filename***

Explanation: sshd refused access to user *username* because the user's rhosts file *filename* is writable by others, or is not owned by the user.

System action: The program continues.

System programmer response: Instruct the user to correct the file modes and/or ownership.

FOTS2174 Authentication refused: *errortext*

Explanation: sshd refused access to a user because the user's authorized keys file, or some component of the pathname, is not secure. The text *errortext* explains further the cause of the problem.

System action: The program continues.

System programmer response: Instruct the user to take action based on *errortext*.

FOTS2175 Nasty PTR record "*name***" is set up for** *ipaddr***, ignoring**

Explanation: When sshd performed a reverse lookup for *ipaddr*, it received a numeric hostname *name*. sshd will use the IP address rather than the returned hostname.

System action: The program continues.

System programmer response: Verify that the entries in the Domain Name System (DNS) database are correct.

FOTS2176 reverse mapping checking getaddrinfo for *hostname* **failed – POSSIBLE BREAKIN ATTEMPT!**

Explanation: When sshd attempted to map *hostname* back to an IP address, a call to getaddrinfo() failed. sshd will use the socket IP address rather than the returned hostname from the Domain Name System (DNS) server.

System action: The program continues.

System programmer response: Verify that the entries in the Domain Name System (DNS) database are correct.

FOTS2177 Address *ipaddr* **maps to** *hostname***, but this does not map back to the address – POSSIBLE BREAK–IN ATTEMPT!**

Explanation: When sshd attempted to map *hostname* back to an IP address using DNS, the returned IP address *ipaddr* differed from that associated with the socket. sshd will use the socket IP address rather than the returned hostname from the Domain Name System (DNS) server.

System action: The program continues.

System programmer response: Verify that the entries in the Domain Name System (DNS) database are correct.

FOTS2178 Connection from *ipaddr* **with IP options:***options*

Explanation: A call to getsockopt() failed for the IP address *ipaddr* with options *options*.

System action: The program ends.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2179 Invalid command.

Explanation: The ssh user attempted to open a command line using the escape character with "C". Only -L and -R (to add port forwardings) are supported commands, but the user entered something else.

System action: The program continues.

User response: Only use the -L or -R options with the command line escape.

FOTS2180 Not supported for SSH protocol version 1.

Explanation: The ssh user attempted to open a command line and specify local port forwarding (using -L) using the escape character with "C". This is not supported for SSH Protocol Version 1.

System action: The program continues.

User response: Use -L in an open command line with SSH Protocol Version 2.

FOTS2181 Bad forwarding port(s)."

Explanation: One of the port numbers specified with ssh options -R or -L are invalid. A port number should be greater than zero and less than or equal to 65535.

System action: The program continues.

User response: Reissue **ssh** with valid port numbers.

FOTS2182 Port forwarding failed.

Explanation: ssh was unable to set up port forwarding. Another error message describes the problem.

System action: The program continues.

User response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2183 User *username* **not allowed because** *progname* **exists**

Explanation: User *username* was not allowed to log in because the nologin program, *progname*, exists.

System action: The program ends.

System programmer response: None.

FOTS2184 You don't exist, go away!

Explanation: A call to getpwuid() failed for the current running user id.

System action: The program ends.

User response: Follow local procedures for reporting problems to IBM.

FOTS2185 Packet integrity error (*length* bytes remaining) at *filename*:*linenum*

Explanation: An internal error occurred.

System action: The program ends.

User response: Follow local procedures for reporting problems to IBM.

FOTS2186 tcgetattr: *error_message*

Explanation: The tcgetattr() system call failed. The daemon is unable to set the terminal modes for the child session. The system error is displayed with the message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2187 Setting tty modes failed: *system error*

Explanation: A call to tcsetattr() failed. The daemon is unable to set the terminal modes for the child session.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2188 *type* host key for IP address '*ipaddr*' not in list of known hosts.

Explanation: ssh found the user has an old-style user known_hosts file, known_hosts2, and checked that file for the host key for *ipaddr*. ssh was unable to find the host key of type *type* for *ipaddr*. The IP address is being checked because CheckHostIP is enabled.

System action: The program continues.

User response: Verify you really meant to use the known_hosts2 file. If so, add the correct host key for *ipaddr*. It is possible the host key just changed.

FOTS2189 Failed to add the *type* host key for IP address '*ipaddr*' to the list of known hosts (*hostfile*).

Explanation: ssh attempted to add the host key for *ipaddr* to the user hostfile *hostfile*, but failed. The host key attempted is of type *type*. The IP address is being checked because CheckHostIP is enabled.

System action: The program continues.

User response: Verify that the user hostfile *hostfile* is writable by the user.

FOTS2190 Failed to add the host to the list of known hosts (*hostfile*).

Explanation: ssh detected a new host key and attempted to add it to the user hostfile *hostfile*, but failed.

System action: The program continues.

User response: Verify that the user hostfile *hostfile* is writable by the user.

FOTS2191 WARNING: Encryption is disabled! Password will be transmitted in clear text.

Explanation: The user is using ssh with Protocol Version 1 and password authentication. ssh detected a cipher is not getting used for encryption. This should not occur, since in Protocol Version 1 if "none" is specified, 3des should be used.

System action: The program continues.

User response: Follow local procedures for reporting problems to IBM.

FOTS2192 Warning: privilege separation user should not be UID 0.

Explanation: The privilege separation user (SSHD) is defined to be UID 0, but it should be defined to an unprivileged (non-UID 0) user ID. Defining this user as UID 0 may decrease the effectiveness of privilege separation. This may also cause problems with some security products.

System action: The program continues.

System programmer response: Redefine the SSHD privilege separation user to be a non-UID 0 user ID.

FOTS2193 Failed to change code sets to convert between "*from_codeset*" and "*to_codeset*".

Explanation: The OpenSSH daemon attempted to change the internal code sets used for data conversion. This occurs if the remote process changes the code sets of the terminal. For example, a user issuing the **chcp**

command from the remote shell could initiate this processing.

System action: The daemon will continue to use the previous setting for data conversion. The program continues.

User response: Verify that conversion is possible between the code sets specified by the user. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2194 __tcgetcp() failed: *system error*

Explanation: A call to __tcgetcp() failed while **sshd** was trying to obtain the code set information for the master pty. The system error is displayed with this message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2195 *function* **failed :** *system error*

Explanation: A call to *function* failed. The system error is displayed with this message.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2196 iconv failed. Conversion stopped at 0x*hexbyte***. System Error:** *system error*

Explanation: A call to iconv() failed indicating that a byte did not have a representation in the destination codeset. Conversion failed at byte *hexbyte*. The system error is displayed with this message.

System action: The program continues.

User response: Verify that conversion is possible between the code sets specified by the user. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2197 *function_name***: read only partial extended packet data. len:***bytes* **data:***packet flag* **System Error:***system error*

Explanation: A call to read() expected at least four bytes of extended packet data and received only *bytes* bytes, shown in *packet flag*. If an application attempted to change the code sets for the allocated terminal, this action may not have been performed. The system error is displayed with this message.

System action: The program continues.

User response: Verify that conversion is possible between the code sets specified by the user. If applicable, reissue the **chcp** command. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2198 kexgex_client: BN_bin2bn failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2199 X11 connection rejected because of wrong authentication.

Explanation: An X11 connection has been rejected because of incorrect authentication information.

System action: The program continues.

User response: Verify that the authentication information for the X11 connection is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2201 ssh_kex: BN_set_word failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2202 ssh_kex: BN_lshift failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2203 ssh_kex: BN_add_word failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2204 ssh: connect to host *host_name* **port** *port*: *error_message*

Explanation: Connection to host *host_name* on port *port* could not be established. The system error is displayed with the message.

System action: The program continues.

User response: Verify that a server is listening for connections on the specified host and port, and try the request again. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2205 Connection timed out during banner exchange

Explanation: The connection timed out while exchanging banner information.

System action: The program ends.

User response: Verify that a server is listening for connections on the specified host and port, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2206 ssh_exchange_identification: select: *error_message*

Explanation: The select() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2207 ssh_exchange_identification: No banner received

Explanation: The connection failed to complete the banner exchange. No banner was received.

System action: The program ends.

User response: Verify that a server is listening for connections on the specified host and port, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2208 Tunnel forwarding is disabled to avoid man–in–the–middle attacks.

Explanation: Strict host key checking (refer to the ssh_config StrictHostKeyChecking keyword) has not been requested, so the connection is allowed, but tunnel forwarding is disabled.

System action: The program continues.

User response: The ssh_config Tunnel keyword is not supported on z/OS UNIX. Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config keywords. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2209 Couldn't execute *shell_path* **–c "***shell_arguments***":** *error_message*

Explanation: The execl() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2210 Couldn't wait for child: *error_message*

Explanation: The waitpid() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2211 PRIV_START: seteuid: *error_message*

Explanation: The seteuid() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error. Also, verify that the **ssh** command has the noshareas extended attribute set.

FOTS2212 PRIV_END: seteuid: *error_message*

Explanation: The seteuid() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error. Also, verify that the ssh program has the noshareas extended attribute set. The attribute can be set via the **extattr** command.

FOTS2213 Warning: No xauth data; using fake authentication data for X11 forwarding.

Explanation: Unable to generate xauth key data for X11 forwarding. Fake data will be used.

System action: The program continues.

User response: Verify that the location of the xauth program is valid and that the program is capable of generating the required xauth key data, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config XAuthLocation keyword.

FOTS2214 Timeout, server not responding.

Explanation: The **ssh** sesssion ended because the server did not respond within the time allowed. The number of server alive messages sent exceeded the value set by the ssh_config ServerAliveCountMax keyword.

System action: The program ends.

User response: Verify that the server is active, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config ServerAliveCountMax keyword.

FOTS2215 Could not request tunnel forwarding.

Explanation: The tunnel forwarding request has failed.

System action: The program ends.

User response: Tunnel forwarding is not supported on z/OS UNIX. Remove the tunnel forwarding request, and try again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on tunnel forwarding. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2216 Could not request tunnel forwarding.

Explanation: The tunnel forwarding request has failed.

System action: The program continues.

User response: Tunnel forwarding is not supported on z/OS UNIX. Remove the tunnel forwarding request, and try again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on tunnel forwarding. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2217 Error: remote port forwarding failed for listen port *listen_port*

Explanation: A remote forwarding request failed for listen port *listen_port*.

System action: The program ends.

User response: The server failed to complete the remote forwarding request. Verify that the remote forwarding request is valid on the server, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2218 ControlPath too long

Explanation: The control path is too long.

System action: The program ends.

User response: Verify that the control path is valid, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config ControlPath keyword. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2219 *function* **socket():** *error_message*

Explanation: The socket() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2220 Not supported.

Explanation: Cancel local forwarding –KL is not a supported **ssh** command line option.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2221 Bad forwarding close port

Explanation: Bad port specified for the –KR **ssh** command line option.

System action: The program continues.

User response: Verify that a valid port is specified, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2222 Bad forwarding specification.

Explanation: Bad forwarding specification for a **ssh** command line option.

System action: The program continues.

User response: Verify that a valid forwarding specification was specified, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2224 ControlSocket *control_path* **already exists**

Explanation: The control socket for the control path *control_path* already exists.

System action: The program ends.

User response: Verify that the control path does not exist, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config ControlPath keyword. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2225 *function* **bind():** *error_message*

Explanation: The bind() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2226 client_input_channel_req: request for channel –1

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2227 client_input_channel_req: unexpected channel *session_id*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2228 Warning: untrusted X11 forwarding setup failed: xauth key data not generated

Explanation: Untrusted X11 forwarding could not be set up because xauth key data could not be generated.

System action: The program continues.

User response: Verify that the location of the xauth program is valid and that the program is capable of generating the required xauth key data, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config XAuthLocation keyword.

FOTS2229 *function***: no channel for id** *channel_id*

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2230 Request failed on channel *channel_id*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2231 *function*: **cctx == NULL**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2232 *function* **accept:** *error_message*

Explanation: The accept() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2233 *function* **getpeereid failed:** *error_message*

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2234 control mode uid mismatch: peer euid *peer_effective_UID* **!= uid** *real_UID*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2235 *function*: **client msg_recv failed**

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2236 *function*: **wrong client version** *version*

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2237 *function*: **client msg_send failed**

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2238 Unsupported command *command_value*

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2239 Refused control connection

Explanation: Internal error.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2240 client_session2_setup: channel *channel_id*: **unknown channel**

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2241 *function*: **failed to receive fd** *file_descriptor* **from slave**

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2242 *function*: **tcgetattr:** *error_message*

Explanation: The tcgetattr() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2243 Tunnel forwarding is not supported for protocol 1

Explanation: Tunnel forwarding is not supported for SSH protocol version 1.

System action: The program continues.

User response: The ssh_config Tunnel keyword is not supported on z/OS UNIX. Remove the keyword from the ssh_config file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config Tunnel keyword. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2244 Tunnel device open failed.

Explanation: The tunnel device failed to open.

System action: The program continues.

User response: The ssh_config Tunnel keyword is not supported on z/OS UNIX. Remove the keyword from the ssh_config file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config Tunnel keyword. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2245 *filename* **line** *line_number*: **Bad number.**

Explanation: The value for the ssh_config keyword in file *filename* at line *line_number* contains a bad number.

System action: The program ends.

User response: Verify that the value for the ssh_config keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config keywords. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2246 *filename* **line** *line_number*: **Invalid RekeyLimit suffix**

Explanation: The ssh_config RekeyLimit keyword in file *filename* at line *line_number* is set to a value that contains an invalid suffix.

System action: The program ends.

User response: Verify that the value for the ssh_config RekeyLimit keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config RekeyLimit keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2247 *filename* **line** *line_number*: **RekeyLimit too large**

Explanation: The ssh_config RekeyLimit keyword in file *filename* at line *line_number* is set to a value that is too large.

System action: The program ends.

User response: Verify that the value for the ssh_config RekeyLimit keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config RekeyLimit keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2248 *filename* **line** *line_number*: **RekeyLimit too small**

Explanation: The ssh_config RekeyLimit keyword in file *filename* at line *line_number* is set to a value that is too small.

System action: The program ends.

User response: Verify that the value for the ssh_config RekeyLimit keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config RekeyLimit keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2249 *filename* **line** *line_number*: **missing address family.**

Explanation: The ssh_config AddressFamily keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

User response: Verify that a value for the ssh_config AddressFamily keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config AddressFamily keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2250 *filename* **line** *line_number*: **Invalid environment name.**

Explanation: The sshd_config SendEnv keyword in file *filename* at line *line_number* is set to a value that contains an invalid environment variable name.

System action: The program ends.

User response: Verify that the value for the ssh_config SendEnv keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config SendEnv keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2251 *filename* **line** *line_number*: **too many send env.**

Explanation: Too many environment variables have been specified by the ssh_config SendEnv keywords.

System action: The program ends.

User response: Verify that the ssh_config SendEnv keywords do not specify too many environment variables, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config SendEnv keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2252 *filename* **line** *line_number*: **Missing ControlMaster argument.**

Explanation: The ssh_config ControlMaster keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

User response: Verify that a value for the ssh_config ControlMaster keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config ControlMaster keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2253 *filename* **line** *line_number*: **Bad ControlMaster argument.**

Explanation: The ssh_config ControlMaster keyword in file *filename* at line *line_number* is set to an unsupported value.

System action: The program ends.

User response: Verify that the value for the ssh_config ControlMaster keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config ControlMaster keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2254 *filename* **line** *line_number*: **Missing yes/point–to–point/ethernet/no argument.**

Explanation: The ssh_config Tunnel keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

User response: The ssh_config Tunnel keyword is not supported on z/OS UNIX. Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config Tunnel keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2255 *filename* **line** *line_number*: **Bad yes/point–to–point/ethernet/no argument:** *value*

Explanation: The ssh_config Tunnel keyword in file *filename* at line *line_number* is set to an unsupported value *value*.

System action: The program ends.

User response: The ssh_config Tunnel keyword is not supported on z/OS UNIX. Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config Tunnel keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2256 *filename* **line** *line_number*: **Bad tun device.**

Explanation: The ssh_config TunnelDevice keyword in file *filename* at line *line_number* is set to an unsupported value.

System action: The program ends.

User response: The ssh_config TunnelDevice keyword is not supported on z/OS UNIX. Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the ssh_config TunnelDevice keyword. If unable to resolve, contact your system programmer.

System programmer response: If file *filename* refers to the system–wide ssh_config file then correct the error in the file, and have the user try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2257 **fstat** *filename*: *error_message*

Explanation: The fstat() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2258 **Bad owner or permissions on** *filename*

Explanation: The owner or access permissions on file *filename* are set to values that are not secure.

System action: The program ends.

User response: Verify that you own the file and that write access permission is only granted to the owner, and try the request again.

FOTS2259 **Commands:**

Explanation: Help was requested for the **ssh** command line options.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2260 **–L[bind_address:]port:host:hostport Request local forward**

Explanation: Help was requested for the **ssh** command line options.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2261 **–R[bind_address:]port:host:hostport Request remote forward**

Explanation: Help was requested for the **ssh** command line options.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2262 **–KR[bind_address:]port Cancel remote forward**

Explanation: Help was requested for the **ssh** command line options.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2263 !args Execute local command

Explanation: Help was requested for the **ssh** command line options.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** command line options.

FOTS2264 No support for tunnel device forwarding.

Explanation: The **ssh** –w option is not supported on z/OS UNIX.

System action: The program continues.

User response: Verify that the **ssh** –w option is not specified, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** –w option.

FOTS2265 Warning: Could not request remote forwarding.

Explanation: A remote forwarding request has failed.

System action: The program continues.

User response: Check for additional error messages displayed with this message, and take appropriate action. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the error messages displayed with this message. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2266 Warning: remote port forwarding failed for listen port *listen_port*

Explanation: A remote forwarding request failed for listen port *listen_port*.

System action: The program continues.

User response: The server failed to complete the remote forwarding request. Verify that the remote forwarding request is valid on the server, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2267 Pseudo–terminal will not be allocated because stdin is not a terminal.

Explanation: A pseudo–terminal will not be allocated because stdin is not a terminal.

System action: The program continues.

User response: If a pseudo–terminal must be allocated then use the **ssh** –t option to force the allocation of a pseudo–terminal. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the **ssh** –t option.

FOTS2268 Warning: Remote host refused compression.

Explanation: The compression request sent to the server failed or was denied.

System action: The program continues.

User response: Verify that the server is set up to allow compression, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2269 Warning: Remote host failed or refused to allocate a pseudo tty.

Explanation: The pseudo tty request sent to the server failed or was denied.

System action: The program continues.

User response: Verify that the server is set up to allow pseudo tty allocation, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2270 Warning: Remote host denied X11 forwarding.

Explanation: The X11 forwarding request sent to the server failed or was denied.

System action: The program continues.

User response: Verify that the server is set up to allow X11 forwarding, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2271 Warning: Remote host denied authentication agent forwarding.

Explanation: The agent forwarding request sent to the server failed or was denied.

System action: The program continues.

User response: Verify that the server is set up to allow

agent forwarding, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2272 Agent forwarding disabled for protocol 1.3

Explanation: Agent forwarding not supported with SSH protocol version 1.3.

System action: The program continues.

User response: Use SSH protocol version 2, and try the request again.

FOTS2273 Warning: Permanently added the *key_type* host key for IP address '*ip_address*' to the list of known hosts.

Explanation: The *key_type* host key for IP address *ip_address* was added to your known hosts file.

System action: The program continues.

User response: Verify that the added host key matches the server's actual host key. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on setting up server authentication.

FOTS2274 Warning: Permanently added '*host_name*' (*key_type*) to the list of known hosts.

Explanation: The *key_type* host key for host *host_name* was added to your known hosts file.

System action: The program continues.

User response: Verify that the host key added matches the server's actual host key. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on setting up server authentication.

FOTS2275 WARNING: *key_type* key found for host *host_name*

in *filename*:*line_number* *key_type* key fingerprint *key_fingerprint*.

Explanation: The *key_type* host key for host *host_name* was found in file *filename* at line *line_number*.

System action: The program continues.

User response: Verify that the host key found matches the server's actual host key. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on setting up server authentication.

FOTS2276 Warning: the *key_type* host key for '*host_name*' differs from the key for the IP address '*ip_address*'

Offending key for IP in *filename*:*line_number*

Explanation: The host key found for host name *host_name* differs from the key found for IP address *ip_address*. The offending IP address key was found in file *filename* at line *line_number*.

System action: The program continues.

User response: Correct the host keys, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on setting up server authentication.

FOTS2277 Matching host key in *filename*:*line_number*

Explanation: The host key found for the host name differs from the key found for the IP address. The offending host name key was found in file *filename* at line *line_number*.

System action: The program continues.

User response: Correct the host key, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on setting up server authentication.

FOTS2278 *function*: no channel for id *channel_id*

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2279 *function*: stat("*filename*") failed: *error_message*

Explanation: The stat() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2280 *function*: **fstat(***file_descriptor***) failed:** *error_message*

Explanation: The fstat() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2281 *function*: **open("***filename***") failed:** *error_message*

Explanation: The open() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2282 *function*: **open("/dev/zero") not valid**

Explanation: The /dev/zero file opened is not valid. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Verify that the /dev/zero file is a valid character special file. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2283 *function*: **dup2(***file_descriptor1***,** *file_descriptor2***) failed:** *error_message*

Explanation: The dup2() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2306 **User** *user_name* **from** *host_name* **not allowed because listed in DenyUsers**

Explanation: Access denied for user *user_name*. The user was denied access through the sshd_config DenyUsers keyword.

System action: The program continues.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config DenyUsers keyword.

FOTS2307 **User** *user_name* **from** *host_name* **not allowed because not listed in AllowUsers**

Explanation: Access denied for user *user_name*. The user was not listed with the sshd_config AllowUsers keyword.

System action: The program continues.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config AllowUsers keyword.

FOTS2308 **User** *user_name* **from** *host_name* **not allowed because not in any group**

Explanation: Access denied for user *user_name*. The user does not have any groups associated with it.

System action: The program continues.

System programmer response: Follow local procedures for setting up user accounts.

FOTS2309 **User** *user_name* **from** *host_name* **not allowed because a group is listed in DenyGroups**

Explanation: Access denied for user *user_name*. The user belongs to a group that was denied access through the sshd_config DenyGroups keyword.

System action: The program continues.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config DenyGroups keyword.

FOTS2310 **User** *user_name* **from** *host_name* **not allowed because none of user's groups are listed in AllowGroups**

Explanation: Access denied for user *user_name*. The user belongs to groups that were not listed with the sshd_config AllowGroups keyword.

System action: The program continues.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config AllowGroups keyword.

FOTS2311 **expand_authorized_keys: path too long**

Explanation: The pathname for the user's authorized_keys file is too long.

System action: The program ends.

System programmer response: Verify that the value of

| the sshd_config AuthorizedKeysFile keyword is valid. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2312 auth_rsa_generate_challenge: BN_rand failed

Explanation: Internal error.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2313 auth_rsa_generate_challenge: BN_mod failed

Explanation: Internal error.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2314 kexdh_server: BN_bin2bn failed

Explanation: Internal error.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2315 *function*: **Unexpected KEX type** *KEX_type*

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2316 DH_compute_key: failed

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2317 kexgex_server: BN_bin2bn failed

Explanation: Internal error.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2318 *function*: **Cannot find account for uid** *UID*

Explanation: The getpwuid() system call failed to get information about a user with UID *UID*. The failure occurred in *function*.

System action: The program ends.

System programmer response: Verify that the UID is valid. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2319 *function*: **Cannot find user "***user_name***"**

Explanation: The getpwnam() system call failed to get information about user *user_name*. The failure occurred in *function*.

System action: The program ends.

System programmer response: Verify that the user name *user_name* is valid. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2323 *function*: **authentication method name unknown**

Explanation: A client attempted an unknown authentication method. The failure occurred in *function*.

System action: The program ends.

System programmer response: Verify that the client is requesting valid authentication methods. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2324 *function*: **send fds failed**

Explanation: Failed to send terminal file descriptors to the unprivileged child process. The failure occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2326 *function*: **write:** *error_message*

Explanation: The write() system call failed. The system error is displayed with the message. The failure occurred in *function*.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error and take the appropriate action. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2327 *function*: **read:** *error_message*

Explanation: The read() system call failed. The system error is displayed with the message. The failure occurred in *function*.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error and take the appropriate action. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2328 *function*: **option block size mismatch**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2329 *function*: **receive fds failed**

Explanation: Failed to receive terminal file descriptors from the monitor process. The failure occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2330 **listen on [***host_name***]:***port*: *error_message*

Explanation: The sshd daemon failed to listen on port *port*. The listen() system call failed. The system error is displayed with the message.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2331 **reexec socketpair:** *error_message*

Explanation: The socketpair() system call failed. The system error is displayed with the message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2332 *function*: **ssh_msg_send failed**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2333 *function*: **ssh_msg_recv failed**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2334 *function*: **rexec version mismatch**

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2335 **sshd re–exec requires execution with an absolute path**

Explanation: The **sshd** command was called without using an absolute path.

System action: The program ends.

System programmer response: Call the **sshd** command using an absolute path, and try the request again.

FOTS2336 **rexec of** *filename* **failed:** *error_message*

Explanation: The execv() system call failed. The system error is displayed with the message.

System action: The program continues.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2337 **session_x11_req: session** *session_id*: **x11 forwarding already active**

Explanation: The client requested X11 forwarding for session *session_id* when X11 forwarding is already active.

System action: The program continues.

System programmer response: Verify that the client requests X11 forwarding only when it's not already

active. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2338 chroot path does not begin at root

Explanation: The chroot directory pathname does not begin at the current root directory ('/').

System action: The program ends.

System programmer response: Verify that the value of the sshd_config ChrootDirectory keyword is valid, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the sshd_config ChrootDirectory keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2339 chroot path too long

Explanation: The chroot directory pathname is too long.

System action: The program ends.

System programmer response: Verify that the value of the sshd_config ChrootDirectory keyword is valid, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the sshd_config ChrootDirectory keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2340 *function*: **stat("***pathname_component***"):** *error_message*

Explanation: The stat() system call failed. The system error is displayed with the message. The failure occurred in *function* while processing pathname component *pathname_component* of the chroot directory pathname.

System action: The program ends.

System programmer response: Verify that the value of the sshd_config ChrootDirectory keyword is valid, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the sshd_config ChrootDirectory keyword and to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2341 bad ownership or modes for chroot directory *string***"***pathname_component***"**

Explanation: The pathname component *pathname_component* of the chroot directory pathname has incorrect ownership or mode settings.

System action: The program ends.

System programmer response: Verify that the ownership and mode settings of the chroot directory pathname components are valid, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the sshd_config ChrootDirectory keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2342 chroot path *string***"***pathname_component***" is not a directory**

Explanation: The pathname component *pathname_component* of the chroot directory pathname is not a directory.

System action: The program ends.

System programmer response: Verify that all pathname components of the chroot directory pathname are directories, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the sshd_config ChrootDirectory keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2343 Unable to chdir to chroot path "*pathname***":** *error_message*

Explanation: The chdir() system call failed to change the working directory to the chroot directory pathname *pathname*. The system error is displayed with the message.

System action: The program ends.

System programmer response: Verify that the value of the sshd_config ChrootDirectory keyword is valid, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the sshd_config ChrootDirectory keyword. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2344 chroot("*pathname***"):** *error_message*

Explanation: The chroot() system call failed to change the root directory to the chroot directory pathname *pathname*. The system error is displayed with the message.

System action: The program ends.

System programmer response: Verify that the value of the sshd_config ChrootDirectory keyword is valid, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the sshd_config ChrootDirectory keyword. Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2345 *function*: **chdir(/) after chroot:** *error_message*

Explanation: The chdir() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

System programmer response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2346 **session_close_single_x11: no x11 channel** *channel_id*

Explanation: Internal error.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2347 **You must change your password now and login again!**

Explanation: Your password has expired and must be changed.

System action: The program continues.

User response: Change your password and login again.

FOTS2348 *function*: **no message header**

Explanation: No message header found while attempting to receive a file descriptor. The error occurred in *function*.

System action: The program continues.

User response: Try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2349 *filename* **line** *line_number*: **Directive '***keyword***' is not allowed within a Match block**

Explanation: The keyword *keyword* in file *filename* at line *line_number* is not allowed within a Match block specified by the Match keyword.

System action: The program ends.

System programmer response: Verify that the keywords within the Match block are correct, and try the request again. Refer to the OpenSSH daemon configuration files information in *IBM Ported Tools for z/OS User's Guide* for more information on the Match keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2350 *filename* **line** *line_number*: **missing address family.**

Explanation: The sshd_config AddressFamily keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

System programmer response: Verify that a value for the sshd_config AddressFamily keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config AddressFamily keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2351 *filename* **line** *line_number*: **address family must be specified before ListenAddress.**

Explanation: The sshd_config AddressFamily keyword in file *filename* at line *line_number* must be specified before the sshd_config ListenAddress keyword.

System action: The program ends.

System programmer response: Specify the sshd_config AddressFamily keyword before the sshd_config ListenAddress keyword in the file *filename*, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config keywords. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2352 *filename* **line** *line_number*: **unsupported address family "***value***".**

Explanation: The sshd_config AddressFamily keyword in file *filename* at line *line_number* is set to an unsupported value *value*.

System action: The program ends.

System programmer response: Verify that the value for the sshd_config AddressFamily keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config AddressFamily keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2353 *filename* **line** *line_number*: **missing yes/no/delayed argument.**

Explanation: The sshd_config Compression keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

System programmer response: Verify that a value for the sshd_config Compression keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config Compression keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2354 *filename* **line** *line_number*: **Bad yes/no/delayed argument:** *value*

Explanation: The sshd_config Compression keyword in file *filename* at line *line_number* is set to an unsupported value *value*.

System action: The program ends.

System programmer response: Verify that the value for the sshd_config Compression keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config Compression keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2355 *filename* **line** *line_number*: **missing yes/no/clientspecified argument.**

Explanation: The sshd_config GatewayPorts keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

System programmer response: Verify that a value for the sshd_config GatewayPorts keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config GatewayPorts keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2356 *filename* **line** *line_number*: **Bad yes/no/clientspecified argument:** *value*

Explanation: The sshd_config GatewayPorts keyword in file *filename* at line *line_number* is set to an unsupported value *value*.

System action: The program ends.

System programmer response: Verify that the value for the sshd_config GatewayPorts keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config GatewayPorts keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2357 *filename* **line** *line_number*: **Invalid environment name.**

Explanation: The sshd_config AcceptEnv keyword in file *filename* at line *line_number* is set to a value that contains an invalid environment variable name.

System action: The program ends.

System programmer response: Verify that the value for the sshd_config AcceptEnv keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config AcceptEnv keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2358 *filename* **line** *line_number*: **too many allow env.**

Explanation: Too many environment variables have been specified by the sshd_config AcceptEnv keywords.

System action: The program ends.

System programmer response: Verify that the sshd_config AcceptEnv keywords do not specify too many environment variables, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config AcceptEnv keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2359 *filename* **line** *line_number*: **Missing yes/point–to–point/ethernet/no argument.**

Explanation: The sshd_config PermitTunnel keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

System programmer response: The sshd_config PermitTunnel keyword is not supported on z/OS UNIX. Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config PermitTunnel keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2360 *filename* **line** *line_number*: **Bad yes/point–to–point/ethernet/no argument:** *value*

Explanation: The sshd_config PermitTunnel keyword in file *filename* at line *line_number* is set to an unsupported value *value*.

System action: The program ends.

System programmer response: The sshd_config PermitTunnel keyword is not supported on z/OS UNIX. Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config PermitTunnel keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2361 **Match directive not supported as a command–line option**

Explanation: The Match keyword is not supported as a command–line option.

System action: The program ends.

System programmer response: Specify the Match keyword in the appropriate configuration file, and try the request again. Refer to the OpenSSH daemon configuration files information in *IBM Ported Tools for z/OS User's Guide* for more information on the Match

keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2362 *filename* **line** *line_number*: **Bad Match condition**

Explanation: The Match keyword in file *filename* at line *line_number* is set to a bad Match condition.

System action: The program ends.

System programmer response: Verify that the value for the Match keyword is correct, and try the request again. Refer to the OpenSSH daemon configuration files information in *IBM Ported Tools for z/OS User's Guide* for more information on the Match keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2363 *filename* **line** *line_number*: **missing PermitOpen specification**

Explanation: The sshd_config PermitOpen keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

System programmer response: Verify that a value for the sshd_config PermitOpen keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config PermitOpen keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2364 *filename* **line** *line_number*: **missing host in PermitOpen**

Explanation: The sshd_config PermitOpen keyword in file *filename* at line *line_number* is missing the host value.

System action: The program ends.

System programmer response: Verify that the value for the sshd_config PermitOpen keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config PermitOpen keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2365 *filename* **line** *line_number*: **bad port number in PermitOpen**

Explanation: The sshd_config PermitOpen keyword in file *filename* at line *line_number* contains a bad port number.

System action: The program ends.

System programmer response: Verify that the value for the sshd_config PermitOpen keyword is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config PermitOpen keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2366 *filename* **line** *line_number*: **Missing argument.**

Explanation: The sshd_config ForceCommand keyword in file *filename* at line *line_number* is missing its value.

System action: The program ends.

System programmer response: Verify that a value for the sshd_config ForceCommand keyword is set, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config ForceCommand keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2368 **line** *line_number*: **too many groups in Match Group**

Explanation: The Match keyword at line *line_number* contains too many values for the Group Match criteria.

System action: The program continues.

System programmer response: Verify that the value for the Match keyword is correct, and try the request again. Refer to the OpenSSH daemon configuration files information in *IBM Ported Tools for z/OS User's Guide* for more information on the Match keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2369 **Missing Match criteria for** *match_criteria*

Explanation: The Match keyword is missing the value for the Match criteria *match_criteria*.

System action: The program continues.

System programmer response: Verify that a value for the Match keyword is set, and try the request again. Refer to the OpenSSH daemon configuration files information in *IBM Ported Tools for z/OS User's Guide* for more information on the Match keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2370 **Unsupported Match attribute** *value*

Explanation: The Match keyword is set to an unsupported criteria value *value*.

System action: The program continues.

System programmer response: Verify that the criteria value for the Match keyword is correct, and try the request again. Refer to the OpenSSH daemon configuration files information in *IBM Ported Tools for z/OS User's Guide* for more information on the Match keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2371 permanently_set_uid: no user given

Explanation: Internal error.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2372 password change not supported

Explanation: A user requested a password change during password authentication. The password change is not supported.

System action: The program continues.

System programmer response: Inform the user that a password change must be requested after password authentication.

FOTS2373 wrong user name passed to monitor: expected *expected_user_name* **!=** *user_name*

Explanation: The wrong user name *user_name* was passed to the monitor process during authentication. The monitor process expected user name *expected_user_name*.

System action: The program continues.

System programmer response: Verify that the client passed a valid user name. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2374 *filename* **line** *line_number***: Deprecated option** *keyword*

Explanation: The keyword *keyword* in file *filename* at line *line_number* is no longer supported.

System action: The program continues.

System programmer response: Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config keywords. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2375 *filename* **line** *line_number***: Unsupported option** *keyword*

Explanation: The keyword *keyword* in file *filename* at line *line_number* is not supported.

System action: The program continues.

System programmer response: Remove the keyword from the file, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config keywords. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2376 subsystem request for *subsystem* **failed, subsystem not found**

Explanation: Subsystem request failed. The subsystem *subsystem* was not found.

System action: The program continues.

System programmer response: Verify that the subsystem requested by the client is valid and is supported by the sshd_config Subsystem keyword. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config Subsystem keyword. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2377 Disabling protocol version 1. Could not load host key

Explanation: Protocol version 1 was disabled because one or more host keys could not be loaded.

System action: The program continues.

System programmer response: Verify that a host key for protocol version 1 exists. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config HostKey keyword. Host keys specified by the HostKeyRingLabel keyword are not supported for protocol 1. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2378 Disabling protocol version 2. Could not load host key

Explanation: Protocol version 2 was disabled because one or more host keys could not be loaded.

System action: The program continues.

System programmer response: Verify that a host key specification for protocol version 2 exists. Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the HostKey and HostKeyRingLabel keywords. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2379 Attempt to write login records by non–root user (aborting)

Explanation: The **sshd** daemon attempted to write login records under a user with a UID not equal to zero.

System action: The program continues.

System programmer response: Verify that the **sshd** daemon was started with a user with a UID of zero.

FOTS2380 *function*: **utmp_write_library() failed**

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2381 *function*: **invalid type field**

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2382 **Warning:** *filename*, **line** *line_number*: **keysize mismatch: actual** *actual_keysize* **vs. announced** *announced_keysize*.

Explanation: The keysize *announced_keysize* on line *line_number* in file *filename* is incorrect. The correct keysize is *actual_keysize*.

System action: The program continues.

System programmer response: Correct the keysize, and try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2383 **Timeout, client not responding.**

Explanation: The number of client alive messages sent without response from the client exceeded the threshold set by the sshd_config ClientAliveCountMax keyword.

System action: The program ends.

System programmer response: Refer to *IBM Ported Tools for z/OS User's Guide* for more information on the sshd_config ClientAliveCountMax keyword.

FOTS2384 *function*: **open("/dev") failed:** *error_message*

Explanation: The open() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2385 *function*: **Unable to set the controlling tty.**

Explanation: The controlling tty could not be set because /dev/tty is not accessible. The error occurred in *function*.

System action: The program continues.

System programmer response: Verify that SSH protocol version 2 is being used, and try the request again. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2386 *function*: **fchdir(***file_descriptor***) failed:** *error_message*

Explanation: The fchdir() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2387 *function*: **chdir("***filename***") failed:** *error_message*

Explanation: The chdir() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2388 *function*: **stat("***filename***") failed:** *error_message*

Explanation: The stat() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2389 *function*: **stat("***filename***") mismatch:** *expected_st_ino actual_st_ino expected_st_dev actual_st_dev*

Explanation: The stat() system call returned unexpected stat information. The error occurred in *function*.

System action: The program ends.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2390 *function*: **close(***file_descriptor***) failed:** *error_message*

Explanation: The close() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

System programmer response: Take appropriate action based on the system error. If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2401 **do_local_cmd: no arguments**

Explanation: Internal error. No arguments for the local command.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2402 **do_local_cmd: fork:** *error_message*

Explanation: The fork() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2403 **do_local_cmd: waitpid:** *error_message*

Explanation: The waitpid() system call failed. The system error is displayed with the message.

System action: The program ends.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2502 *function*: **offset < 0**

Explanation: Internal error. Unexpected file offset was calculated. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2701 *filename* **line** *line_number*: **keyword** *keyword* **is not allowed in file** *filename*.

Explanation: The z/OS–specific keyword *keyword* can not be specified in file *filename*.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for information about *keyword*, and try the request again.

FOTS2702 *filename* **line** *line_number*: **missing** *keyword* **value.**

Explanation: The keyword *keyword* in file *filename* at line *line_number* is missing its value.

System action: The program ends.

User response: Verify that the value for *keyword* is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the *keyword* keyword. If unable to resolve, contact your system programmer.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2703 *filename* **line** *line_number*: **unsupported** *keyword* **value '***value***'.**

Explanation: The keyword *keyword* in file *filename* at line *line_number* is set to an unsupported value *value*.

System action: The program ends

User response: Verify that the value for *keyword* is correct, and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for more information about the *keyword* keyword. If unable to resolve, contact your system programmer.

System programmer response: If unable to resolve, follow local procedures for reporting problems to IBM.

FOTS2704 *filename1* **line** *line_number*: **keyword** *keyword* **is only allowed in file** *filename2*.

Explanation: The z/OS–specific keyword *keyword* can only be specified in the file *filename2*.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for information about *keyword*, and try the request again.

FOTS2705 *filename* **line** *line_number*: **keyword** *keyword* **is not allowed in a z/OS–specific configuration file.**

Explanation: The keyword *keyword* is not a valid z/OS–specific client configuration keyword.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid z/OS client configuration keywords, and try the request again.

FOTS2707 *function*: *system_call*: *system_error*

Explanation: The *system_call* call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2708 *filename* **line** *line_number*: **keyword** *keyword* **is not allowed in a z/OS–specific per–user client configuration file**

Explanation: The keyword *keyword* can not be specified in file *filename*.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for information about *keyword*, and try the request again.

FOTS2709 *file_name* **line** *line_number*: *keyword* **value** *value* **requires additional system setup.**

Explanation: The support provided by *keyword value* requires additional system setup.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for information on setting up OpenSSH to collect SMF records.

FOTS2710 *function*: *callable_service* **failed with message number** *number*.

Explanation: Language Environment callable service failed. The error occurred in *function*.

System action: The program continues.

User response: Refer to *z/OS Language Environment Programming Reference* for an explanation of the message number. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2711 *filename* **line** *line_number*: **keyword** *keyword* **is not allowed in a z/OS–specific daemon configuration file.**

Explanation: The keyword *keyword* is not a valid z/OS–specific daemon configuration keyword.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for valid z/OS–specific daemon configuration keywords, and try the request again.

FOTS2801 *function*: **No SMF data received from master process.**

Explanation: The master process of the specified multiplexed connection did not send the requested SMF data.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2802 *function*: **Error writing SMF record:** *system_error*

Explanation: Failure occurred while writing an SMF record.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2803 *function*: **Error collecting SMF data.**

Explanation: Failure occurred while collecting data for an SMF record. The SMF record will not be written.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2804 *function*: **Error collecting SMF data for** *field_name*.

Explanation: Failure occurred while collecting SMF record data for the specified field. The SMF record will be written without valid data for that field.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2805 *function*: **Bad request size for SMF data length** *actual_data_length*, **expected** *expected_data_length*.

Explanation: Communication error occurred while collecting data for an SMF record. The SMF record will not be written.

System action: The program ends.

User response: Verify connectivity and remote host status. If error persists, contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2806 *function*: **unexpected server login failure reason.**

Explanation: An unexpected server login failure reason was identified. The problem occurred in *function*.

System action: The program continues.

User response: None.

FOTS2807 *function*: **bad SMF global data length** *actual_data_length*, **expected** *expected_data_length*.

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2808 *function*: **unexpected SMF error for type** *SMF_record_type*, **subtype** *SMF_record_subtype* **record:** *error_message*.

Explanation: The __smf_record2() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: SMF records will not be recorded. The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2809 *function*: **bad authentication method** *authentication_method*.

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2810 *function*: **unable to resolve pathname** *pathname* **during SMF data collection:** *error_message*.

Explanation: The realpath() system call failed. The SMF data may not contain an absolute pathname. The system error is displayed with the message. The error occurred in *function*.

System action: The program continues.

User response: Refer to *z/OS XL C/C++ Run-Time Library Reference* for an explanation of the system error. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the system error.

FOTS2811 *function*: **Incorrect SMF** *request_type value*.

Explanation: Internal error. The error occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2812 *function*: **Unknown** *option value*.

Explanation: Internal error. The error occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2813 *function*: **Incorrect data length** *length* **read from SMF pipe.**

Explanation: Failure occurred while collecting data for an SMF record. The SMF record will not be written.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2814 *function*: **ClientSMF keyword value** *value* **requires additional system setup.**

Explanation: The support provided by the zos_ssh_config file keyword ClientSMF *value* requires additional system setup.

System action: SMF records will not be recorded. The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for information on setting up OpenSSH to collect SMF records.

FOTS2815 *function*: **Caller not permitted to use __smf_record2(): ***error_message*.

Explanation: The __smf_record2() system call failed. The system error is displayed with the message. The error occurred in *function*.

System action: The program ends.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for information on what you need to verify before using OpenSSH. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2816 *function*: **__smf_record2() system call not supported.**

Explanation: The __smf_record2() system call is not supported. Additional system setup is required to use this system call. The error occurred in *function*.

System action: The program continues.

User response: Refer to *IBM Ported Tools for z/OS User's Guide* for information on what you need to verify before using OpenSSH. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2817 *function*: **Pathname** *pathname* **with resolved directory pathname** *dirname* **is too long.**

Explanation: Unable to resolve the pathname. The resulting pathname is too long. The SMF data may not contain an absolute pathname.

System action: The program continues.

User response: Verify that the pathname is correct, and try the request again. If unable to resolve, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2818 *function*: **Received SMF status** *status1*, **expected** *status2*.

Explanation: An unexpected SMF status value was read. The value does not match the SMF status set in the z/OS–specific client configuration file. The problem occurred in *function*.

System action: The program ends.

User response: Verify connectivity and ssh server status. If unable to resolve, contact your system programmer to report the problem.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2901 *function*: **RSA_new failed**

Explanation: Internal error. The failure occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2902 *function*: **BN_bin2bn failed on** *component*

Explanation: Internal error. The failure occurred in *function*.

System action: The program ends.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2903 *function*: **RSA_blinding_on failed**

Explanation: Internal error. The failure occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2904 *function*: **gsk_factor_public_key_rsa failed (***return_code***).** *return_code_description*.

Explanation: The gsk_factor_public_key_rsa() system call failed when trying to read an RSA public key associated with a certificate in a key ring. The failure occurred in *function*. The *return_code_description* indicates the problem with the certificate.

System action: The program continues.

User response: If more information is needed about the error, refer to *z/OS Cryptographic Services System SSL*

Programming for an explanation of the return code. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the return code.

FOTS2905 *function*: **gsk_factor_private_key_rsa failed (***return_code***).** *return_code_description*.

Explanation: The gsk_factor_private_key_rsa() system call failed when trying to read an RSA private key associated with a certificate in a key ring. The failure occurred in *function*. The *return_code_description* indicates the problem with the certificate.

System action: The program continues.

User response: If more information is needed about the error, refer to *z/OS Cryptographic Services System SSL Programming* for an explanation of the return code. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the return code.

FOTS2906 *function*: **d2i_DSAparams on public key failed**

Explanation: The d2i_DSAparams() system call failed when trying to read a DSA public key associated with a certificate in a key ring. The failure occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2907 *function*: **ASN1_item_d2i on** *key_usage* **key failed**

Explanation: The ASN1_item_d2i() system call failed when trying to read a DSA key component from a key associated with a certificate in a key ring. The failure occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2908 *function*: **unexpected algorithm ID** *algorithm_ID*, **key ring '***key_ring***' label '***certificate_label***'**

Explanation: The algorithm type of the keys associated with the certificate is neither RSA nor DSA. The failure occurred in *function*.

System action: The program continues.

User response: Contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2909 *function*: **Value '***value***' is not valid, leading double quote not found**

Explanation: Either the format of the value is not correct, or unmatched double quotes were found in the string. The failure occurred in *function*.

System action: The program continues.

User response: Correct the value and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for information on the correct format when specifying a key ring or certificate label.

FOTS2910 *function*: **Certificate label found when not expecting one in '***value***'**

Explanation: The value should only contain a key ring identification. The failure occurred in *function*.

System action: The program continues.

User response: Correct the value and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for information on the correct format when specifying a key ring.

FOTS2911 *function*: **Certificate label is missing but is required in '***value***'**

Explanation: The value should contain a key ring identification followed by a certificate label. The failure occurred in *function*.

System action: The program continues.

User response: Correct the value and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for information on the correct format when specifying a key ring and certificate label.

FOTS2912 *function*: **Could not get key from key ring '***key_ring***' label '***certificate_label***'**

Explanation: A valid key could not be extracted from the certificate. The failure occurred in *function*.

System action: The program continues.

User response: Verify that the certificate label correctly identifies a valid certificate and try the request again. There may be other error messages preceding this message that provide more details about the problem. If unable to resolve the problem, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2913 *function*: **Could not get all keys from key ring '***key_ring***'**

Explanation: Valid keys could not be extracted from the certificates associated with *key_ring*. The failure occurred in *function*.

System action: The program ends.

User response: Verify that the key ring correctly identifies the key ring containing valid certificates with keys to be used on this **ssh–add** request and try the request again. There may be other error messages preceding this message that provide more details about the problem. If unable to resolve the problem, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2914 *function*: **Certificate validation for key ring '***key_ring***' label '***certificate_label***' failed (***return_code***).** *return_code_description***.**

Explanation: The *return_code_description* indicates the problem with the certificate. If more information is needed about the error, then refer to *z/OS Cryptographic Services System SSL Programming* for an explanation of the return code. If unable to resolve, contact your system programmer. The failure occurred in *function*.

System action: The program continues if a key is found in a different certificate available to the program.

User response: Correct the condition causing the certificate to fail validation, and try the request again. If unable to resolve the problem, contact your system programmer.

System programmer response: Follow local procedures for reporting problems to IBM.

FOTS2915 *function*: **gsk_open_keyring on '***key_ring***' failed (***return_code***).** *return_code_description***.**

Explanation: The gsk_open_keyring() system call failed when trying to open the key ring. The failure occurred in *function*. The *return_code_description* indicates the problem.

System action: The program continues.

User response: If more information is needed about the error, refer to *z/OS Cryptographic Services System SSL Programming* for an explanation of the return code. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the return code.

FOTS2916 *function*: **gsk_get_record_by_label from key ring '***key_ring***' for label '***certificate_label***' failed (***return_code***).** *return_code_description***.**

Explanation: The gsk_get_record_by_label() system call failed when trying to obtain the data base record for the certificate. The failure occurred in *function*. The *return_code_description* indicates the problem.

System action: The program continues.

User response: If more information is needed about the error, refer to *z/OS Cryptographic Services System SSL Programming* for an explanation of the return code. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the return code.

FOTS2917 *function*: **gsk_get_record_by_index from key ring '***key_ring***' for index '***record_index***' failed (***return_code***).** *return_code_description***.**

Explanation: The gsk_get_record_by_index() system call failed when trying to obtain the data base record for the certificate. The failure occurred in *function*. The *return_code_description* indicates the problem.

System action: The program continues.

User response: If more information is needed about the error, refer to *z/OS Cryptographic Services System SSL Programming* for an explanation of the return code. If unable to resolve, contact your system programmer.

System programmer response: Take appropriate action based on the return code.

FOTS2918 *function*: **Value '***value***' is not valid, trailing double quote was found**

Explanation: Either the format of the value is not correct, or unmatched double quotes were found in the string. The failure occurred in *function*.

System action: The program continues.

User response: Correct the value and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide* for information on the correct format when specifying a key ring or certificate label.

FOTS2919 *function*: **Value '***value***' is not valid, trailing double quote not found**

Explanation: Either the format of the value is not correct, or unmatched double quotes were found in the string. The failure occurred in *function*.

System action: The program continues.

User response: Correct the value and try the request again. Refer to *IBM Ported Tools for z/OS User's Guide*

for information on the correct format when specifying a key ring or certificate label.

FOTS2920 *function*: **Private key not available for certificate in key ring 'key_ring' with label 'certificate_label'**

Explanation: Either there is no private key associated with the certificate, or the user is not authorized to read the private key for the certificate. Only a certificate owner may read the private key for a certificate. The failure occurred in *function*.

System action: The program continues.

User response: If the program obtained a private key from another certificate, then this error may be ignored. If the program did not obtain a private key, then an alternate certificate needs to be specified when trying the request again.

Appendix A. Accessing MVS data sets within sftp

OpenSSH's **sftp** does not have built-in support for MVS data sets. However, there are alternate (indirect) ways to access MVS data sets within **sftp**.

Solution 1: From within **sftp**, use a shell escape to copy between MVS and the z/OS UNIX file system. Do this by preceding any shell command by a '!'.

Example:

```
!cp "//'CTWARE.C(HELLO)'" hello.c
```

The 'HELLO' member is copied to a local file `hello.c`, which could then be transferred from **sftp**. This would be executed while you are within an sftp shell

Note: The `hello.c` file will remain in the z/OS UNIX file system until it is manually removed.

You can use this solution from within an **sftp** batch file as well, to automate certain tasks or help in removal of the file:

```
> cat batchfile
lcd sftptest
cd Test
!cp "//'CTWARE.C(HELLO)'" hello.c
put hello.c
!rm hello.c
> sftp -b batchfile user@remotehost
```

This example would change directories (both local and remote), copy an MVS dataset to the z/OS UNIX file system (on the local machine), transfer the file (to the remote system), and then remove the (local) z/OS UNIX file system copy. This would save you some work, and you would not have to manually remove 'temporary' files.

Tip: Because the **sftp** exit value is not affected by shell command escapes, Solution 2 is preferred if verification of a successful copy is required.

Solution 2: Copy the data from an MVS dataset to the z/OS UNIX file system prior to using **sftp**.

Example:

```
cp "//'CTWARE.C(HELLO)'" hello.c
```

The 'HELLO' member is copied to a local file `hello.c`, which could then be transferred from **sftp**. This would be executed from a standard z/OS UNIX shell

Note: The `hello.c` file remains in the z/OS UNIX file system until it is manually removed.

Appendix B. OpenSSH - port forwarding examples

OpenSSH - without TCP forwarding

Direct client/server connection (no forwarding)

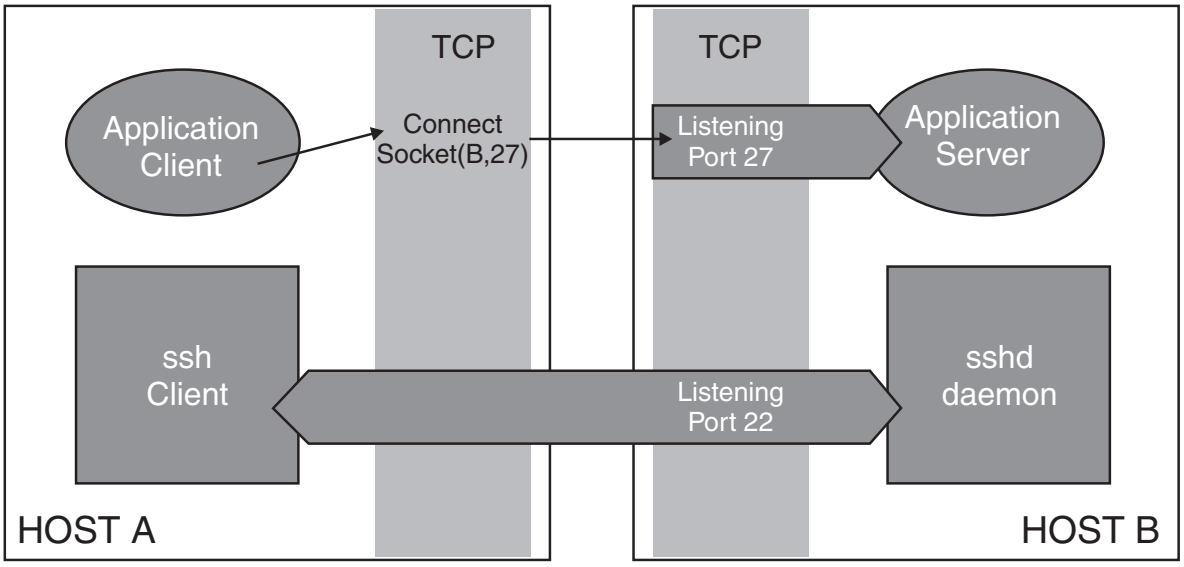

Figure 7. OpenSSH - without TCP port forwarding

OpenSSH - with TCP port forwarding

OpenSSH provides TCP port forwarding, also known as tunnelling, which allows other TCP applications to forward their network data over a secure SSH connection. In other words, existing TCP applications that do not encrypt their data before sending it across the network can send their network traffic through an SSH channel, thereby securing it.

Without TCP forwarding, an application's client connections directly to its server across the network, as shown in Figure 7. To use port forwarding, an existing SSH session must exist.

Example: An example of invoking the **ssh** client to support local port forwarding is:

```
ssh -L 2001:remotehost:27 billy@remotehost
```

Result: The **ssh** client on Host A listens on port 2001 for connections (see Figure 8 on page 336). The TCP application will now connect to port 2001 on the local host (Host A), rather than connect to its well-known port on Host B, where the remote server is listening. This is demonstrated in Figure 9 on page 336. The **ssh** client accepts the connection on port 2001 and forwards the application's data to the OpenSSH server (**sshd**) on Host B. **sshd** then forwards the data to the application's well-known port on Host B, as specified on invocation of the **ssh** client to be port 27. This is demonstrated in Figure 10 on page 337.

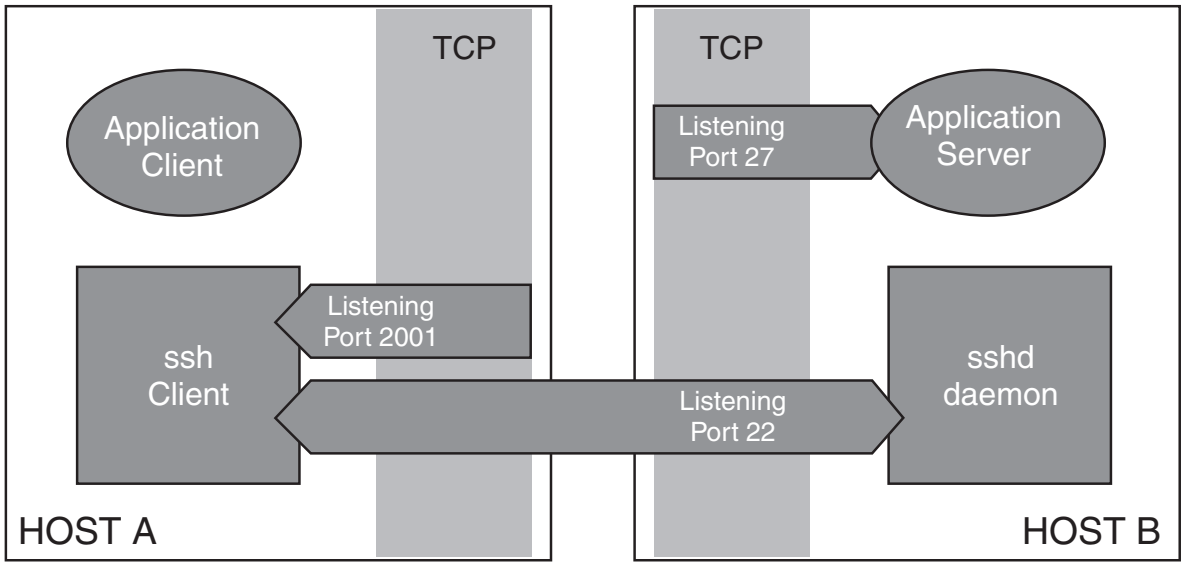

Figure 8. The ssh client is listening on port 2001 for a connection

The TCP application wants to contact the server through a SSH connection.

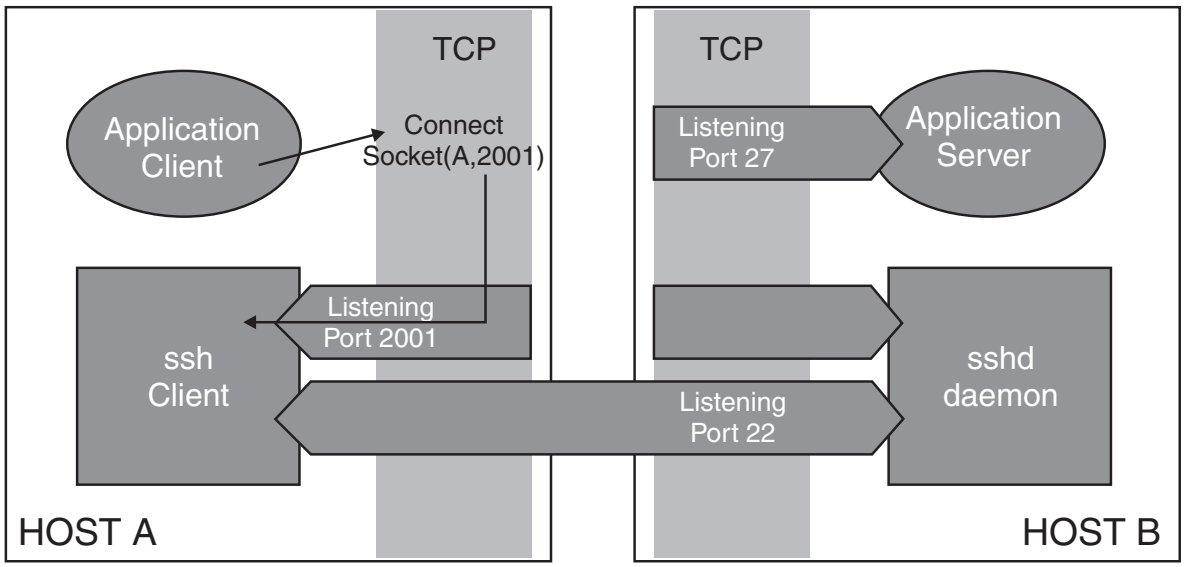

Figure 9. The application is connecting to port 2001 on the local host (Host A)

ssh forwards the data through an SSH tunnel; **sshd** delivers to server.

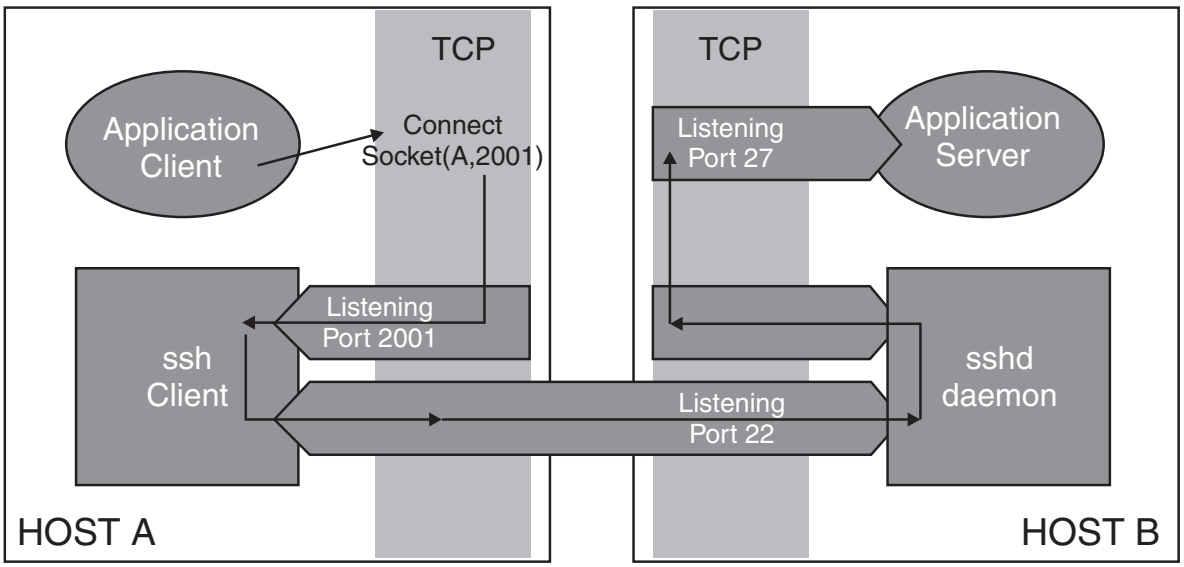

Figure 10. The ssh client accepts the connection on port 2001, forwards the application's data to sshd on Host B, sshd then forwards the data to the application's server, listening on Port 27

Appendix C. RFCs and Internet drafts

The Internet Engineering Task Force (http://www.ietf.org/) has a Secure Shell (SECSH) working group whose goal is to update and standardize the popular SSH protocol. The following SECSH RFCs describe some of the different layers of the protocol:

- The Secure Shell (SSH) Protocol Assigned Numbers, RFC 4250, 2006.
- The Secure Shell (SSH) Protocol Architecture, RFC 4251, 2006.
- The Secure Shell (SSH) Authentication Protocol, RFC 4252, 2006.
- The Secure Shell (SSH) Transport Layer Protocol, RFC 4253, 2006.
- The Secure Shell (SSH) Connection Protocol, RFC 4254, 2006.
- Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints, RFC 4255, 2006.
- Generic Message Exchange Authentication for the Secure Shell Protocol (SSH), RFC 4256, 2006.
- The Secure Shell (SSH) Session Channel Break Extension, RFC 4335, 2006.
- The Secure Shell (SSH) Transport Layer Encryption Modes, RFC 4344, 2006.
- Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol, RFC 4345, 2006.
- Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol, RFC 4419, 2006.
- The Secure Shell (SSH) Public Key File Format, RFC 4716, 2006.

Because internet drafts can be updated, replaced, or obsoleted by newer versions, OpenSSH may only conform to a particular version of the draft. Refer to the IETF Web site at http://www.ietf.org/ for a list of drafts.

Appendix D. Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you may view the information through the z/OS Internet Library Web site or the z/OS Information Center. If you continue to experience problems, send an e-mail to mhvrcfs@us.ibm.com or write to:

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer or Library Server versions of z/OS books in the Internet library at:

`http://www.ibm.com/systems/z/os/zos/bkserv/`

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs that use the OpenSSH portion of Ported Tools for z/OS.

Trademarks

IBM and the IBM logo are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in the OpenSSH portion of the IBM Ported Tools for z/OS documentation. If you do not find the term you are looking for, view IBM Glossary of Computing Terms, located at: http://www.ibm.com/ibm/terminology

A

address space identifier (ASID). A unique, system-assigned identifier for an address space.

ASID. See address space identifier.

B

Basic Encoding Rules (BER). A set of rules used to encode Abstract Syntax Notation One (ASN.1) values as strings of octets.

BCD. See binary-coded decimal.

BER. See Basic Encoding Rules.

binary-coded decimal (BCD). A system for encoding decimal numbers in binary form to avoid rounding and conversion errors. In BCD, the digits of a decimal number are individually represented in 4-bit binary notation. For example, the decimal number 1024 is recorded in BCD as 0001000000100100.

C

CERT Coordination Center (CERT/CC). The CERT/CC is a major reporting center for Internet security problems. Staff members provide technical advice and coordinate responses to security compromises, identify trends in intruder activity, work with other security experts to identify solutions to security problems, and disseminate information to the broad community. The CERT/CC also analyzes product vulnerabilities, publishes technical documents, and presents training courses. For more detailed information about the CERT/CC, see "Meet the CERT/CC" at http://www.cert.org/meet_cert/meetcertcc.html.

CERT/CC. See CERT Coordination Center (CERT/CC).

certificate. In computer security, a digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority and is digitally signed by that authority.

certificate authority. An organization that issues digital certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use, and revokes certificates belonging to users who are no longer authorized to use them.

D

Data Encryption Standard (DES). A cryptographic algorithm designed to encrypt and decrypt data using a private key.

DER. See Distinguished Encoding Rules.

DES. See Data Encryption Standard.

DH-GEX. See Diffie-Hellman Group Exchange.

Diffie-Hellman Group Exchange (DH-GEX). A key agreement method that allows two parties to derive a shared secret key securely over an open (unprotected) network.

digital certificate. A digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority.

digital signature algorithm (DSA). A security protocol that uses a pair of keys (one public and one private) and a one-way encryption algorithm to provide a robust way of authenticating users and systems. If a public key can successfully decrypt a digital signature, a user can be sure that the signature was encrypted using the private key.

Distinguished Encoding Rules (DER). A standard, based on the Basic Encoding Rules, that is designed to ensure a unique encoding of each ASN.1 value, defined in ITU-T X.690.

DSA. See digital signature algorithm.

F

Federal Information Processing Standard (FIPS). A standard produced by the National Institute of Standards and Technology when national and international standards are nonexistent or inadequate to satisfy the U.S. government requirements.

FIPS. See Federal Information Processing Standard.

G

Generic Security Services Application Programming Interface (GSS-API). An Internet Standard protocol (R2078) that specifies calling conventions by which an application (typically another communication protocol) can obtain authentication, integrity, and confidentiality security services independently of the underlying security mechanisms and technologies, thus allowing the application source code to be ported to different environments.

globalization. In computing, the provision of a single software solution that has (1) multicultural support and (2) a user interface and documentation that is available in one or more languages.

GSS-API. See Generic Security Services Application Programming Interface.

I

IETF. See Internet Engineering Task Force.

Internet Engineering Task Force (IETF). The task force of the Internet Architecture Board (IAB) that is responsible for solving the short-term engineering needs of the Internet. The IETF consists of numerous working groups, each focused on a particular problem. Specifications proposed as standards typically undergo a period of development and review before they are adopted as standards.

invariant character set. A set of characters, such as the syntactic character set, having the same code point assignments in all coded character sets or code pages using a given encoding scheme.

K

Kerberos. The security system of Massachusetts Institute of Technology's (MIT) Project Athena. It uses symmetric key cryptography to provide security services to users in a network.

key. In computer security, a sequence of symbols that is used with a cryptographic algorithm for encrypting or decrypting data. See also private key, public key.

key pair. In computer security, a public key and a private key. The sender uses the private key to encrypt the message. The recipient uses the public key to decrypt the message. Because the private key holds more of the encryption pattern than the public key does, the key pair is called asymmetric.

key ring. In computer security, a file that contains public keys, private keys, trusted roots, and certificates.

M

message authentication code (MAC). In computer security, a value that is a part of a message or accompanies a message and is used to determine that the contents, origin, author, or other attributes of all or part of the message are as they appear to be.

MAC. See message authentication code.

MTU. See maximum transmission unit.

multilevel security. A security policy that allows the classification of data and users based on a system of hierarchical security levels (for example: unclassified, secret, top secret) combined with a system of non-hierarchical security categories (for example: Project A, Project B, Project C). The system imposes mandatory access controls restricting which users can access data based on a comparison of the classification of the users and the data. In order to access data, a user must have a security level greater than or equal to that of the data, and be authorized to all of the categories assigned to the data. The mandatory access controls exist in addition to any discretionary access controls (such as access lists) that users can manipulate, and a user must pass both the mandatory controls and any discretionary controls in order to access the data protected by those controls.

maximum transmission unit (MTU). The largest possible unit of data that can be sent on a given physical medium in a single frame. For example, the maximum transmission unit for Ethernet is 1500 bytes.

P

PAM. See Pluggable Authentication Module.

Pluggable Authentication Module (PAM). A programming interface that enables third-party security methods to be used. PAM enables multiple types of authentication, such as Kerberos and the Rivest-Shamir-Adleman (RSA) algorithm, to be used without changing login services.

passphrase. A type of password that is used to control access to OpenSSH authentication keys. It typically contains a sequence of words, punctuation, numbers, white space, or any string of characters, with a mix of uppercase and lowercase letters, numbers, and nonalphanumeric characters.

password phrase. A string consisting of mixed-case letters, numbers, and special characters, including blanks, that is used to control access to data and systems.

private key. In secure communication, an algorithmic pattern used to encrypt messages that only the corresponding public key can decrypt. The private key is also used to decrypt messages that were encrypted

by the corresponding public key. The private key is kept on the user's system and is protected by a password. See also key, public key.

public key. In secure communication, an algorithmic pattern used to decrypt messages that were encrypted by the corresponding private key. A public key is also used to encrypt messages that can be decrypted only by the corresponding private key. Users broadcast their public keys to everyone with whom they must exchange encrypted messages. See also key, private key.

R

Rivest-Shamir-Adleman algorithm (RSA). A public-key encryption technology developed by RSA Data Security, Inc, and used in the IBM implementation of SSL.

RSA. See Rivest-Shamir-Adleman algorithm.

S

SAF. See System Authorization Facility.

seed. A value that adds randomness to the creation of pseudorandom numbers.

Secure Sockets Layer (SSL). A security protocol that provides communication privacy. With SSL, client/server applications can communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

SMF. See System Management Facilities.

SOCKS server. A proxy server that provides a secure one-way connection through a firewall to server applications in a nonsecure network. The server applications in the secure network must be compatible with the socket interface.

SSL. See Secure Sockets Layer.

System Authorization Facility (SAF). A z/OS interface with which programs can communicate with an external security manager, such as RACF.

System Management Facilities (SMF). A component of z/OS that collects and records a variety of system and job-related information.

T

TLS. See Transport Layer Security.

Transport Layer Security. An Internet Engineering Task Force (IETF)-defined security protocol that is based on Secure Sockets Layer (SSL) and is specified in RFC 2246.

Index

Special characters

A

B

C

N

O

P

R

S

T

U

V

W

X

Z

Program Number: 5655-M23

Printed in USA

SA23-2246-00