z/OS

IBM Ported Tools for z/OS User's Guide

SA22-7985-00

z/OS

IBM Ported Tools for z/OS User’s Guide

SA22-7985-00

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 169.

First Edition, May 2004

This edition applies to Version 1 Release 4 of z/OS (5694-A01), to Version 1 Release 4 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers’ comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

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

Contents

About this document

This document presents the information you need to set up and use the OpenSSH client.

Who should use this document?

This document is for the 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. On other open systems, some system programmer tasks may be done by an administrator.

This document assumes the readers are familiar with z/OS systems and with the information for z/OS 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*.

Direct your request for copies of any IBM publication to your IBM representative or to the IBM branch office serving your locality.

There is also a toll-free customer support number (1-800-879-2755) available Monday through Friday from 6:30 a.m. through 5:00 p.m. Mountain Time. You can use this number to:

- Order or inquire about IBM publications
- Resolve any software manufacturing or delivery concerns
- Activate the program reorder form to provide faster and more convenient ordering of software updates.

Softcopy publications

IBM Ported Tools for z/OS User's Guide is available on the *z/OS Software Products Collection*, SK3T-4270. This softcopy collection contains libraries for multiple releases for software products that run on z/OS. All documents are provided in BookManager format and in PDF format, when available. The CD-ROM collection includes the IBM Softcopy Reader and Adobe Reader for reading the softcopy documents.

Softcopy z/OS publications are also 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 www.ibm.com/servers/eserver/zseries/zos/bkserv/.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

```
http://www.ibm.com/servers/resourcelink
```

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, and VSE:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX System Services running OMVS).
- Your Microsoft Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Microsoft Windows DOS command line.
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

1. z/OS.e™ customers received a Memo to Licensees, (GI10-0684) that includes this key code.

IBM Systems Center publications

IBM systems centers produce redbooks that can be helpful in setting up and using z/OS UNIX System Services. You can order these publications through normal channels, or you can view them with a web browser. See the s/390 Redbooks site at www.ibm.com/redbooks.

These documents have not been subjected to any formal review nor have they been checked for technical accuracy, but they represent current product understanding (at the time of their publication) and provide valuable information on a wide range of z/OS UNIX topics. You must order them separately. A selected list of these documents is on the z/OS UNIX web site at http://www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1pub.html/.

z/OS UNIX home page

The z/OS UNIX home page on the World Wide Web has the latest technical news, customer stories, tools, and FAQs (frequently asked questions). You can visit it at www.ibm.com/servers/eserver/zseries/zos/unix/.

Some of the tools available from the web site are ported tools, and some are home-grown tools designed for z/OS UNIX. All this code works in our environment at the time we make it available, but is not officially supported. Each tool has a README file that describes the tool and any restrictions on its use.

The simplest way to reach these tools is through the z/OS UNIX home page. From the home page, click on **Tools and Toys**.

The code is also available from **ftp://ftp.software.ibm.com/s390/zos/unix/** through **anonymous ftp**.

Restrictions

Because the tools are not officially supported, APARS cannot be accepted.

Discussion list

A mailing list (discussion list) that is not sponsored by IBM may 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/

Customers and IBM participants also discuss z/OS UNIX on the **mvs-oe discussion list**. This list is not operated or sponsored by IBM.

To subscribe to the mvs-oe discussion so you can receive postings, send a note to:

`listserv@vm.marist.edu`

Include the following line in the body of the note, substituting your first name and last name as indicated:

```
subscribe mvs-oe first_name last_name
```

After you are subscribed, you will receive further instructions on how to use the mailing list.

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

IBM Ported Tools for z/OS contains a ported application, OpenSSH. OpenSSH provides secure encryption for both remote login and file transfer. Some of the utilities that it includes are:

- **ssh**, a client program for logging into a z/OS shell. 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 implementation of **sshd** supports both SSH protocol versions 1 and 2 simultaneously.

  The default **sshd** configuration runs only 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, IBM Ported Tools for z/OS uses algorithms such as Blowfish and 3DES.

Migrating from downloaded versions

If you are using a downloaded version of OpenSSH, such as the version that can be obtained from the Tools and Toys page on the z/OS UNIX System Services (z/OS UNIX) web site, you need to perform the following steps to migrate to the version of OpenSSH shipped in IBM Ported Tools for z/OS.

The Tools and Toys section is at www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html.

Steps for migrating from a downloaded version

Before you begin: You need to determine if you have a downloaded version on your system.

1. Since there are many different sources where you might have downloaded a version of OpenSSH, use the **find** or **whence** commands to determine if any of the following programs exist on your system:

   ssh
   sshd
   scp
   sftp
   sftp-server
   ssh-add
   ssh-agent
   ssh-keygen
   ssh-keyscan
   ssh-keysign
   ssh-rand-helper
   ssh-askpass

   Remove these programs or move them to a backup directory. Now you can continue with the installation of OpenSSH provided in the IBM Ported Tools for z/OS.

2. Compare configuration files to IBM-provided samples, which may have different default values, and modify, if necessary. Refer to Step 2 on page 4 and Step 3 on page 4 for more details.

3. Keep existing host key files, known hosts files, authorized key files, and user files.

4. Go through the steps outlined in “Setting up the sshd daemon” on page 3. Some of the steps may not be applicable to your particular situation.

When you are done, you have migrated to the version of OpenSSH in IBM Ported Tools for z/OS.

Chapter 2. For system administrators: Setting up and starting the sshd daemon

Overview of setting up and starting the sshd daemon

Before the system administrator can start the **sshd** daemon, setup tasks must be done; the tasks are explained in "Setting up the sshd daemon." Information about configuring the system for X11 forwarding is also provided.

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.

Restriction: OpenSSH does not run in multibyte locales.

The steps in this chapter assume that IBM Ported Tools for z/OS has been installed, or that you have migrated from an downloaded version as described in "Steps for migrating from a downloaded version" on page 1.

In this chapter

This chapter covers the following subtasks.

Subtasks	Associated procedure (see . . .)
Creating or editing the configuration file	"Steps for creating or editing configuration files"
Performing setup for server authentication	"Steps for performing setup for server authentication" on page 6
Creating the **sshd** privilege separation user	"Step for creating the sshd privilege separation user" on page 7
Starting the **sshd** daemon	"Steps for starting sshd as a stand-alone daemon" on page 8 or "Steps for starting the sshd daemon under inetd" on page 9
Configuring the system for X11 forwarding	"Steps for configuring the system for X11 forwarding" on page 10

Setting up the sshd daemon

You must perform certain tasks before you can start the **sshd** daemon:

- Create or edit configuration files.
- Perform setup for server authentication.
- Create the **sshd** privilege separation user.

Steps for creating or editing configuration files

Before you begin: You must make sure that certain directories were set up correctly when z/OS UNIX was installed:

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 7.
/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 – OpenSSH daemon configuration files" on page 63.
/etc/ssh	755	UID(0)	Holds the configuration files for **ssh** and **sshd**.

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. They must be stored in the IBM-1047 (EBCDIC) code set.

```
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
```

Table 1 describes the permission and UID settings for each configuration file.

Table 1. Configuration files

File	Copied to	Description	Permissions	Owner
/samples/sshd_config	/etc/sshd/sshd_config	Configuration settings for the **sshd** daemon.	644	UID(0)
/samples/ssh_config	/etc/ssh/ssh_config	Configuration settings for the**ssh** client.	644	UID(0)
/samples/moduli	/etc/ssh/moduli	Diffie-Hellman groups	644	UID(0)
/samples/ssh_prng_cmds	/etc/ssh/ssh_prng_cmds	Commands for gathering entropy	644	UID(0)

---

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 — OpenSSH daemon" on page 45 and "sshd_config – OpenSSH daemon configuration files" on page 63.

---

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 — OpenSSH client (remote login program)" on page 22 and "ssh_config – OpenSSH client configuration files" on page 56.

Notes:

a. The settings in this configuration file provide system defaults. They can be overridden by the user's **ssh** configuration in **$HOME/.ssh/config** 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.

Guideline: Do not map multiple MVS identities to the same z/OS UNIX UID, especially for interactive login sessions. However, UID(0) is likely to be shared by multiple MVS identities, and if multiple MVS identities are mapped to the same z/OS UNIX UID, the user may have difficulties running the SSH client. The difficulties occur because the home directory that is retrieved by the SSH client (by looking up the UID in the user database), and used to locate certain user-specific files, is not necessarily the home directory of the current user. To avoid problems when running as a user that shares an UID, a user-specific **ssh_config** file needs to be created, with special attention to setting the IdentityFile and UserKnownHostsFile fields to the proper user-specific values. The user should then always specify this configuration file with the **-F** option when running the SSH client.

---

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.

   Guideline: Run the **sshd** server from a superuser with permission to the BPX.DAEMON FACILITY class profile. If you do so, it can work with low ports restricted.

   You can configure **sshd** to listen on a different port with the Port keyword or the **-p** command-line option (see “sshd_config – OpenSSH daemon configuration files” on page 63).

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

   ```
   Port 1022
   ```

---

5. Set up random number generation. The sample file copied into **/etc/ssh/ssh_prng_cmds** should provide enough entropy for most installations. To produce random numbers, the **ssh** 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.

   Tip: To provide more randomness, you can add more commands to the **/etc/ssh/ssh_prng_cmds** file. However, this may degrade your **ssh** performance.

---

6. Create an **sshrc** file. (This is an optional step.) 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. Set up the OpenSSH message catalog. Change the NLSPATH environment variable in the system-wide shell profiles so that the OpenSSH message catalog will be used. Specifically, in **/etc/profile** and **/etc/csh.login**, edit NLSPATH to include the following setting:

```
/usr/lib/nls/msg/%L/%N.cat
```

Example: If NLSPATH is currently set to:

```
NLSPATH=/usr/lib/nls/msg/%L/%N
```

Change it to:

```
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat
```

When you are done, you have either created or edited a configuration file.

Steps for performing setup for server authentication

Before you begin: You need to know whether you want to use SSH protocol version 1 or version 2. For more information about those protocols, see “SSH protocol version 1” on page 45 and “SSH protocol version 2” on page 46.

Perform the following steps to perform setup for server authentication.

1. Generate the host keys for the SSH server. (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 the IBM-1047 code set.

For SSH Protocol . . .	Then issue . . .
Version 1	`ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_key -N ""`

Version 2	`ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key -N ""`
	 `ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ""`

---

2. Create local and remote **ssh_known_hosts** files. The **ssh_known_hosts** file allows a client to verify a remote host’s identity.
   a. Copy the local host’s public keys to the remote hosts.
      1) Log into your remote host.
      2) Create or edit the remote **/etc/ssh/ssh_known_hosts** file by appending the following local host’s public keys to the **/etc/ssh/ssh_known_hosts** file:
         - **/etc/ssh/ssh_host_key.pub**
         - **/etc/ssh/ssh_host_dsa_key.pub**
         - **/etc/ssh/ssh_host_rsa_key.pub**
      3) 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.
      4) For each public key added to this file, add the hostname of the key to the start of the line. See “SSH_KNOWN_HOSTS file format” on page 50 for more information.
      5) Log off the system.

Clients logging into your host can now verify its identity.

Rule: Update the **ssh_known_hosts** files on remote systems every time host keys are regenerated.

b. Gather the public **ssh** host keys of remote hosts. You can do this by using the **ssh-keyscan** command. Redirect the **ssh-keyscan** output to a file so that you can review the file and verify the keys before adding them to create the **/etc/ssh/ssh_known_hosts** file. If you do not verify the keys before creating **ssh_known_hosts**, users may be vulnerable to attacks.

When you are done, you have performed setup for server authentication.

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 unused group ID and unused nonzero user ID that you want to use.

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
```

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

Starting the sshd daemon

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

- As a stand-alone daemon, as described in “Steps for starting sshd as a stand-alone daemon” on page 8. 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 9. 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

What you need to know before you begin

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

Setting up the z/OS UNIX level of security

For more information about the z/OS UNIX level of security, see the section on establishing the correct level of security for daemons in *z/OS UNIX System Services Planning*.

1. Assign a user ID to the daemon. The user ID must have a UID of 0 and ACCESS(READ) permission to BPX.DAEMON. This user ID may already have been set up on your system.

   Example: The following example assumes that the OMVSKERN 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 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(OMVSKERN) ACCESS(READ)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```
2. The **sshd** daemon is installed with the program control and noshareas extended attributes. If you have not already done so for daemon support, activate Program Control. You may also need to ensure the Language Environment run-time library is defined to program control, as shown in the following example.

   Example:

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

   For more information about program control, see the section on defining programs from load libraries to program control in *z/OS UNIX System Services Planning*.

Steps for starting sshd as a stand-alone daemon

There are several ways to start and restart **sshd**. The method used depends on the level of control the installation has chosen for daemons. You can start daemons in any of these ways:

- **Put the command in /etc/rc 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 **/etc/rc**, 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 **/etc/rc**, 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 with UID(0) and READ permission to BPX.DAEMON.
- **Start sshd with a cataloged procedure using BPXBATCH to invoke a daemon program located in the HFS.** 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 /usr/sbin/sshd -f /etc/ssh/sshd_config'
   //* STDIN and STDOUT are both defaulted to /dev/null
   //STDERR   DD   PATH='/tmp/sshd.stderr',PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
   //         PATHMODE=(SIRWXU)
   ```

   The JCL in the SSHD PROC invokes BPXBATCH, which sets up the standard file descriptors before running **/usr/sbin/sshd**.
2. For this **sshd** catalogued procedure to obtain control with superuser and daemon authority, you must add it to the STARTED class.

   Example: This example assumes that the OMVSKERN user ID is defined as UID(0), and has READ access to the BPX.DAEMON profile in the FACILITY class. For more information on how to set up OMVSKERN, see the section on preparing RACF in *z/OS UNIX System Services Planning*. Following is an example of a catalogued procedure:

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

   For more information about using started procedures and the STARTED class, see the section on using started procedures in *z/OS Security Server RACF Security Administrator's Guide*.
3. To start **sshd**, issue the following command from the MVS console:

   ```
   S SSHD
   ```

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

- **If you are running with UNIX-level security, 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*.

Tip: If the server configuration file is changed after the **sshd** daemon is running, the changes do not affect the daemon, unless the a SIGHUP signal is sent to the daemon process. To restart the **sshd** daemon, reading the configuration file, without terminating existing SSH connections, issue

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

SIGHUP does not reset command-line options (which may override the configuration file). 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 optionally 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   OMVSKERN  /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**. If **inetd** is currently running, send it a SIGHUP signal to allow the new configuration file with **sshd** settings to be read.

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 will only work, 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 **/usr/lpp/tcpip/X11R6/Xamples/clients/xauth** to a local directory.

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

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

2. Edit the Makefile.

   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 xauth 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 may allow users to bypass access restrictions, enabling it is not recommended.

c. Compile the code by issuing **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.

   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 15.

Chapter 3. Getting ready to use OpenSSH

Overview of getting ready to use OpenSSH

This chapter discusses the setup tasks the user must do. It includes the steps for generating user keys, which is a required step, and setting up the system for X11 forwarding, which is an optional step.

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 users on the system might be running in different locales.

Restriction: OpenSSH does not run in multibyte locales.

The steps in this chapter assume that IBM Ported Tools for z/OS has been installed, or that you have migrated from an downloaded version as described in "Steps for migrating from a downloaded version" on page 1.

In this chapter

This chapter covers the following subtasks.

Subtasks	Associated procedure (see . . .)
Setting up the configuration file	"Steps for setting up the configuration file"
Setting up user authentication	"Steps for setting up user authentication" on page 14
Configuring your setup for X11 forwarding	"Steps for configuring your setup for X11 forwarding" on page 15

Steps for setting up the configuration file

Before you begin: You need to know that the user performing these steps should be running in the default (C) locale.

1. Copy the sample configuration file from the **/samples** directory to your **$HOME/.ssh** directory.

   ```
   cp -p /samples/ssh_config  $HOME/.ssh/config
   ```

2. Modify the **$HOME/.ssh/config** file to control the **ssh** client-side authentication methods attempted, protocols and ciphers supported, and session control options. For details, see "ssh — OpenSSH client (remote login program)" on page 22 and "ssh_config – OpenSSH client configuration files" on page 56.

   The settings in this configuration file provide system defaults and can be overridden by command-line options. By prefacing groups of configuration options with the Host keyword, you can share the **ssh_config** file across multiple systems with client configuration options that are tailored to the specific local system being used.

When you are done, you have set up your configuration file.

Steps for setting up user authentication

After user authentication is set up, clients can verify their identities to the server using public key authentication. Public key authentication is the most secure authentication method available in SSH. To use it, minor initial setup is involved, including generating your public and private key pairs, copying public keys to remote hosts, and gathering public keys from other accounts on other hosts.

Before you begin: You need to know which protocol you want to use, SSH Protocol version 1 or SSH protocol version 2. For more information about those protocols, see “SSH protocol version 1” on page 45 and “SSH protocol version 2” on page 46.

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, SSH protocol version 1 or protocol version 2, as shown in Table 2.

   Rule: On z/OS UNIX, these key files must be stored in the IBM-1047 (EBCDIC) code set. Assuming that the user running these commands is running in the default (C) locale, this will occur with no special actions on the part of the user. If you are using a different locale, you do not need to be concerned with the information in this section.

Table 2. Using SSH protocol version 1 and 2

If you are using . . .	Issue . . .
SSH protocol version 1	`ssh-keygen -t rsa1`

SSH protocol version 2	`ssh-keygen -t rsa`
	`ssh-keygen -t dsa`

2. Copy the public keys to all remote hosts that you plan to log in to, using public key authentication. OpenSSH uses the authorized_keys file, by default, to store these public keys.
   a. Log into your remote host.
   b. Create or edit the **$HOME/.ssh/authorized_keys** file for your accounts on both local and remote systems. See Table 3.

Table 3. Creating or editing the $HOME/.ssh/authorized_keys file

If you want to . . .	Then . . .
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 **$HOME/.ssh/authorized_keys** file.
Enable remote users to log into a local account	Append the remote user’s public keys (those ending with a “pub” suffix) to the local user’s **$HOME/.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 move your public key files to another system, treat the files as text to enable any necessary conversion between ASCII and EBCDIC.

   c. Log off the remote system.

When you are done, you have set up user authentication.

Rule: Every time you regenerate the keys, you must update the **authorized_keys** file on remote systems.

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 10.

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 **$HOME/.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 the **ssh** client with the **-X** option. Use this if you want to enable X11 forwarding for this session only.

2. In your local SSH configuration file (**$HOME/.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**).

   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 **$HOME/.ssh/environment** to include **/usr/lib**.

   Example:

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

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

Chapter 4. OpenSSH command descriptions

scp — Secure copy (remote file copy program)

Format

scp [**–pqrvBC46**] [**–F** *ssh_config*] [**–S** *program*] [**–P** *port*] [**–c** *cipher*] [**–i** *identity_file*] [**–o** *ssh_option*] [[*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. Unlike **rcp**, **scp** asks for passwords or passphrases if they are needed for authentication.

Any file name may contain a host and user specification to indicate that the file is to be copied to or from that host. Copies between two remote hosts are permitted. When copying between two remote hosts, only options **-v**, **-r** and **-p** are passed to the remote host regardless of what user specifies on the command line.

scp assumes 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.

Options

–B
: Selects batch mode (prevents asking for passwords or passphrases). To avoid password prompts, use public-key authentication with an **ssh-agent**, host-based authentication or Kerberos if available.

–c *cipher*
: Selects the cipher to use for encrypting the data transfer. This option is directly passed to **ssh**. For more information, see the **–c** option for "ssh — OpenSSH client (remote login program)" on page 22 or the **Ciphers** keyword in "ssh_config – OpenSSH client configuration files" on page 56.

–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**.

–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 — OpenSSH client (remote login program)" on page 22.

–o *ssh_option*
: Can be used to pass options to **ssh** in the format used in **ssh_config**. This is useful for specifying options for which there is no separate **scp** command-line flag.

: **Example:** To use protocol version 1:

```
scp -oProtocol=1
```

–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. This option does not suppress output generated by the **–v** option.

–r
Recursively copy 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.

–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.

Exit values

0	Successful completion
>0	An error occurred.

Related information

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

Authors

Timo Rinne and Tatu Ylonen

sftp — Secure file transfer program

Format

sftp [**–vC1**] [**–b** *batchfile*] [**–o** *ssh_option*] [**–s** *subsystem* | *sftp_server*] [**–B** *buffer_size*] [**–F** *ssh_config*] [**–P** *sftp_server_path*] [**–R** *num_requests*] [**–S** *program*] *host*

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

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

Description

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

sftp connects and logs into the specified *host*, then enters an interactive command mode. The second usage format retrieves files automatically if a non-interactive authentication method is used; otherwise it will do so after successful interactive authentication. The last usage format allows the **sftp** client to start in a remote directory.

By default, **sftp** assumes files are binary. Files copied between EBCDIC and ASCII platforms are not converted. The interactive command ’ascii’ can be used to transfer files in ASCII between local host and remote host.

Options

–b *batchfile*
: Batch mode reads a series of commands from an input batchfile instead of stdin. Because it lacks user interaction, it should be used in conjunction with non-interactive authentication. **sftp** ends if any of the following commands fail: **get**, **put**, **rename**, **ln**, **rm**, **rmdir**, **mkdir**, **cd**, **lcd**, and **lmkdir**.

–B *buffer_size*
: Specifies the size of the buffer that **sftp** uses when transferring files. Larger buffers requires 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 20.

–C
: Enables compression. This option is passed to **ssh**.

–F *ssh_config*
: Specifies an alternate per-user configuration file for **ssh**. This option is directly passed to **ssh**.

–o *ssh_option*
: Can be used to pass options to **ssh** in the format used in the **ssh** configuration file. This is useful for specifying options for which there is no separate **sftp** command-line flag.

  Example: To specify an alternate port, use:

  ```
  sftp -oPort=24
  ```

  For more information, see “ssh_config – OpenSSH client configuration files” on page 56.

–P *sftp_server_path*
: Connects directly to the local **sftp-server** (instead of via **ssh**). This option may be useful in debugging the client and server.

–R *num_requests*
: Specifies the number of requests that can be outstanding at any one time. Increasing this may 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. Maximum is 3.

–1 Specifies the use of protocol version 1. Because 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.

Limitations

Due to limitations in the SECSH protocol with regards to EBCDIC platforms, **sftp** used with OpenSSH protocol version 1 is only supported from z/OS to z/OS.

The biggest buffer size that can be allocated is 2147483647(INT_MAX) bytes. INT_MAX is defined in limits.h.

Interactive commands

Once in interactive mode, **sftp** understands a set of commands similar to those of **ftp**. Commands are case insensitive and path names may be enclosed in quotes if they contain spaces.

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.

This command 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.

chmod *mode path*
Changes permissions of file *path* to *mode*.

chown *own path*
Changes owner of file *path* to *own*. *own* must be a numeric UID.

exit Quits **sftp**.

get [*–P***]** *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. If the **–P** or **–p** flag is specified, then the file's full permission 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* are case sensitive.

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 [–l] [*path***]**
: Displays remote directory listing of either *path* or current directory if *path* is not specified. If the **–l** flag is specified, then displays additional details including permissions and ownership information.

lumask *umask*
: Sets local umask to *umask*.

mkdir *path*
: Creates remote directory specified by *path*.

put [*–P***]** *local-path* **[***remote-path***]**
: Uploads *local-path* and store 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. If the **–P** or **–p** flag is specified, then the file's full permission and access time are copied as well.

pwd Displays remote working directory.

quit Quits **sftp**.

rename *oldpath newpath*
: Renames remote file from *oldpath* to *newpath*.

rmdir *path*
: Removes remote directory specified by *path*.

rm *path*
: Deletes 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 **sftp** version.

! Escapes to local shell.

! *command*
: Executes *command* in local shell.

? Synonym for **help**.

Exit values

0	Successful completion
>0	An error occurred.

Related information

scp, **ssh**, **ssh-add**, **ssh-keygen**, **sftp-server**, **sshd**

Author

Damien Miller

sftp-server — SFTP server subsystem

Format

sftp-server

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 from **sshd** using the *Subsystem* option. See “sshd — OpenSSH daemon” on page 45 for more information.

Related information

sftp, **ssh**, **sshd**

Author

Markus Friedl

ssh — OpenSSH client (remote login program)

Format

ssh [**–l** *login_name*] **hostname** | *user@hostname* [**command**]

ssh [**–afgnqstvxACNTVX1246**] [**–b** *bind_address*] [**–c** *cipher_spec*] [**–e** *escape_char*] [**–i** *identity_file*] [**–l** *login_name*] [**–m** *mac_spec*] [**–o** *option*] [**–p** *port*] [**–F** *configfile*] [**–L** *prt:host:hostport*] [**–R** *port:host:hostport*] [**–D** *port*] *hostname* | *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/IP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified *hostname*. Users must prove their identity to the remote machine using one of several methods depending on the protocol version used :

SSH protocol version 1

First, if the machine the user logs in from is listed in **/etc/hosts.equiv** or **/etc/ssh/shosts.equiv** on the remote machine and the user names are the same on both sides, the user is immediately permitted to log in. Second, if *.rhosts* or *.shosts* exists in the user’s home directory on the remote machine and contains a line containing the name of the client machine and the name of the user on that machine, the user is permitted to log in. This form of authentication alone is normally not allowed by the server because it is not secure. This authentication method is also known as RhostsAuthentication.

The second authentication method is the rhosts or hosts.equiv method combined with RSA-based host authentication. This authentication method is also known as RhostsRSAAuthentication. If the login would be permitted by **$HOME/.rhosts**, **$HOME/.shosts**, **/etc/hosts.equiv**, or **/etc/ssh/shosts.equiv**, and the server can verify the client's host key (see the description for "$HOME/.ssh/known_hosts" on page 30), then the login is permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing.

Note: **/etc/hosts.equiv**, **$HOME/.rhosts**, and the rlogin/rsh protocol in general are inherently insecure and should be disabled if security is an issue.

As a third authentication method, **ssh** supports RSA-based authentication. The scheme is based on public-key cryptography: there are cryptosystems where encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key. RSA is one such system. The idea is that 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. The file **$HOME/.ssh/authorized_keys** lists the public keys that are permitted for logging in. When the user logs in, the **ssh** program tells the server which key pair it would like to use for authentication. The server checks if this key is permitted, and if so, sends the user (actually the **ssh** program running on behalf of the user) a challenge, a random number, encrypted by the user's public key. The challenge can only be decrypted using the proper private key. The user's client then decrypts the challenge using the private key, proving that he/she knows the private key but without disclosing it to the server.

ssh implements the RSA authentication protocol automatically. Users create an RSA key pair by running **ssh-keygen**. Doing so stores the private key in **$HOME/.ssh/identity** and the public key in **$HOME/.ssh/identity.pub**. They should then copy the **identity.pub** to **$HOME/.ssh/authorized_keys** in their home directory on the remote machine (the authorized_keys file corresponds to the conventional **$HOME/.rhosts** file, and has one key per line, though the lines can be very long). They can now log on without giving the password. RSA authentication is much more secure than rhosts authentication.

The most convenient way to use RSA authentication may be with an authentication agent. See "ssh-agent — Authentication agent" on page 35 for more information.

If other authentication methods fail, **ssh** prompts the user for a password. The password is sent to the remote host for checking. However, because all communications are encrypted, the password cannot be seen by someone listening on the network.

SSH protocol version 2

Authentication methods for protocol version 2 are similar to those for protocol version 1. Using the default values for PreferredAuthentications, the client will try authentication methods in the following order until one is successful:

1. Host-based (disabled by default)
2. Public key authentication
3. Keyboard-interactive (not supported on z/OS UNIX)
4. Password authentication

The public key method is similar to RSA authentication described in the previous section and allows the RSA or DSA algorithm to be used: The client uses his private key, **$HOME/.ssh/id_dsa** or **$HOME/.ssh/id_rsa**, to sign the session identifier and sends the result to the server. The server checks whether the matching public key is listed in **$HOME/.ssh/authorized_keys** and grants access if both the key is found and the signature is correct. The session identifier is derived from a shared Diffie-Hellman value and is only known to the client and the server.

If public key authentication fails or is not available, an encrypted password can be sent to the remote host to authenticate the user. Additionally, **ssh** supports host based or challenge response authentication.

Protocol 2 provides additional mechanisms for confidentiality (the traffic is encrypted using 3DES, Blowfish, CAST128, or Arcfour) and integrity (hmac-md5, hmac-sha1). Protocol 1 lacks a strong mechanism for ensuring the integrity of the connection.

Note: Although the documentation for **ssh** often refers to $HOME to mean the current user's home directory, **ssh** does not use the $HOME variable to determine the user's home directory. In the case where multiple MVS identities are mapped to the same UNIX UID, the home directory retrieved by the **ssh** client (by looking up the UID in the user database) is not necessarily the home directory of the current user. To avoid problems when running as a user that shares a UID, a user-specific **ssh_config** file needs to be created, with special attention to setting the IdentityFile and UserKnownHostsFile fields to the proper user-specific values. The user should then always specify this configuration file with the -F option when running the **ssh** client.

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 will be automatically encrypted.

If a pseudo-terminal has been allocated (normal login session), the user may use the escape characters in the next section.

If no pseudo tty 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 pseudo terminal 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 directive or on the command line by the **–e** option.

The supported escapes (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
~C	Open command line (only useful for adding port forwardings using the **–L** and **–R** options)
~R	Request rekeying of the connection (only useful for SSH protocol version 2 and if the peer supports it)

X11 and TCP forwarding

If the **ForwardX11** keyword set to "yes" (or, see the description of the **-X** and **-x** options described in “Options” on page 26) 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 – OpenSSH client configuration files” on page 56.

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.

Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either on the command line or in a configuration file. One possible application of TCP/IP forwarding is a secure connection to an electronic purse; another is going through firewalls.

Server authentication

ssh automatically maintains and checks a database containing identifications for all hosts it has ever been used with. Host keys are stored in **$HOME/.ssh/known_hosts** in the user’s home directory. Additionally, the file **/etc/ssh/ssh_known_hosts** is automatically checked for known hosts. Any new hosts are automatically added to the user’s file. If a host’s identification ever changes, **ssh** warns about this and disables password authentication to prevent a trojan horse from getting the user’s password. Another purpose of this mechanism is to prevent man-in-the-middle attacks which could otherwise be used to circumvent the encryption. The StrictHostKeyChecking option can be used to prevent logins to machines whose host key is not known or has changed.

Options

–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 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*
Specifies the interface to transmit from when using machines with multiple interfaces or aliased addresses. The bind_address must be the same address family (IPv4 or IPv6) as the remote hostname 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 2** specifications, ciphers can be specified in order of preference in a comma-separated list. Valid ciphers include:

3des-cbc
A Triple-DES algorithm

blowfish-cbc
Blowfish algorithm

cast128-cbc
CAST algorithm

arcfour
ARCFOUR algorithm

aes128-cbc
Advanced Encryption Standard (AES) with 128-bit key

aes192-cbc
Advanced Encryption Standard (AES) with 192-bit key

aes256-cbc
Advanced Encryption Standard (AES) with 256-bit key

–C Requests compression of all data (including stdin, stdout, stderr, and data for forwarded X11 and TCP/IP connections). The level can be controlled by the CompressionLevel option. The argument must be an integer from 1 (fast) to 9 (slow, best). The default level is 6, which is good for most applications. Compression is desirable on modem lines and other slow connections, but will decrease performance on fast networks. The default value can be set on a per-host basis in the configuration files; for more information on the Compression and CompressionLevel options see "ssh_config – OpenSSH client configuration files" on page 56.

–D *port*
Specifies a local dynamic application-level port forwarding. This works by allocating a socket to listen to port on the local side and 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 to from the remote machine. Currently, the SOCKS4 protocol is supported and **ssh** will act as a SOCKS4 server. Only a superuser can forward privileged ports. Dynamic port forwardings can also be specified in the configuration file.

–e *ch***|***^ch***|***none*
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 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 configuration file. If a configuration file is given on the command line, the system-wide configuration file (**/etc/ssh/shh_config**) will be ignored. The default for the per user configuration file is **$HOME/.ssh/config**.

–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 **$HOME/.ssh/identity** for protocol version 1 and **$HOME/.ssh/id_rsa** and **$HOME/.ssh/id_dsa** for protocol version 2. Identity files may also be specified on a per-host basis in the configuration file. It is possible to have multiple **–i** options (and multiple identities specified in configuration files).

For a given protocol, identity files are tried in the order they are specified. However, if an identity is loaded in an agent, then that identity is tried first.

–I *smartcard_device*
(**–I** is the uppercase **– i**). It is not supported on z/OS UNIX. Specifies which smartcard device to use. The argument is the device that **ssh** should use to communicate with a smartcard used for storing the user's private RSA key.

–k This is not supported on z/OS UNIX. Disables forwarding of Kerberos tickets and AFS tokens. This option can also be specified on a per-host basis in the configuration file.

–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 configuration file.

–L *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, and whenever 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 configuration file. Only a superuser can forward privileged ports. IPv6 addresses can be specified with an alternative syntax: *port/host/hostport*.

–m *mac_spec*
: For protocol version 2, a comma-separated list of MAC (message authentication code) algorithms can be specified in order of preference. See the definition for “MACs” on page 60 in “Format” on page 56 for more information.

–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 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 configuration file. This is useful for specifying options for which there is no separate command-line flag.

Example:

```
ssh -oHostbasedAuthentication=no Billy@us.pok.ibm.com
```

–p *port*
: Port to connect to on the remote host. This can be specified on a per-host basis in the configuration file.

–q
: Quiet mode. Suppresses all warning and diagnostic messages.

–R *port***:***host***:***hostport*
: Specifies the given port on the remote (server) host is to be forwarded to host and port on the local side. This works by allocating a socket to listen to *port* on the remote side and whenever 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 configuration file. Privileged ports can be forwarded only when logging in as superuser on the remote machine. IPv6 addresses can be specified with an alternative syntax: *port/host/hostport*.

–s
: Can be used to request invocation of a subsystem on the remote system.

Subsystems are a feature of the SSH protocol version 2 which facilitate 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 z/OS. See “Limitations” on page 32 for more information.

–t Forces pseudo-tty allocation. This 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 tty 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 pseudo-tty 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. Maximum is 3.

–V Displays the current OpenSSH and OpenSSL version information and exits.

–x Disables X11 forwarding.

–X Enables X11 forwarding. This can also be specified on a per-host basis in the 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.

–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.

Environment variables set by ssh

ssh will normally set the following environment variables:

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 **$HOME/.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. See the **PermitUserEnvironment** option in "ssh_config – OpenSSH client configuration files" on page 56.

Files

$HOME/.ssh/known_hosts
Records host keys for all hosts the user has logged into that are not in **/etc/ssh/ssh_known_hosts**. See "sshd — OpenSSH daemon" on page 45.

$HOME/.ssh/identity, $HOME/.ssh/id_dsa, $HOME/.ssh/id_rsa
Contains the authentication identity of the user. They are for protocol 1 RSA, protocol 2 DSA, and protocol 2 RSA, respectively. 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.

$HOME/.ssh/identity.pub, $HOME/.ssh/id_dsa.pub, $HOME/.ssh/id_rsa.pub
Contains the public key for authentication (public part of the identity file in human-readable form). The contents of the **$HOME/.ssh/identity.pub** file should be added to **$HOME/.ssh/authorized_keys** on all machines where

the user wishes to log in using protocol version 1 RSA authentication. The contents of the **$HOME/.ssh/id_dsa.pub** and **$HOME/.ssh/id_rsa.pub** file should be added to **$HOME/.ssh/authorized_keys** on all machines where the user wishes to log in using protocol version 2 DSA/RSA authentication. These files are not sensitive and can (but need not) be readable by anyone. These files are never used automatically and are not necessary; they are only provided for the convenience of the user.

$HOME/.ssh/config
The per-user configuration file. The file format and configuration options are described in "ssh_config – OpenSSH client configuration files" on page 56.

$HOME/.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 "sshd — OpenSSH daemon" on page 45. In the simplest form, the format is the same as the **.pub** identity files. This file is not highly sensitive, but recommended permissions are read/write for the user, and not accessible by others.

/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. This file should be world-readable. This file contains public keys, one per line, with the following fields separated by spaces :*system name*, *public key*, and, optionally, *comments*. When different names are used for the same machine, all such names should be listed, separated by commas. For more information on the format, see "sshd — OpenSSH daemon" on page 45.

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 the name servers would then be able to fool host authentication.

/etc/ssh/ssh_config
System-wide configuration file. For file format and configuration information, see "ssh_config – OpenSSH client configuration files" on page 56.

/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 RhostsRSAAuthentication and HostbasedAuthentication. If the protocol version 1 RhostsRSAAuthentication method is used, **ssh** must be **setuid 0**, since the host key is readable only by a superuser. For protocol version 2, **ssh** uses **ssh_keysign** to access the host keys for HostbasedAuthentication. This eliminates the requirement that **ssh** be **setuid 0** when that authentication method is used. By default, **ssh** is not setuid 0.

$HOME/.rhosts

This file is used in .rhosts authentication to list the host/user pairs that are permitted to log in. On many historical UNIX platforms, this file is also used by **rlogin** and **rsh**, which makes using this file insecure. Each line of the file contains a host name in the canonical form returned by name servers and then a user name on that host, separated by a space. 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.

Note that by default, **sshd** is installed so that it requires successful RSA host authentication before permitting .rhosts authentication. If the server machine does not have the client's host key in **/etc/ssh/ssh_known_hosts**, it can be stored in **$HOME/.ssh/known_hosts**. The easiest way to do this is to connect back to the client from the server machine using **ssh**; this will automatically add the host key to **$HOME/.ssh/known_hosts**.

$HOME/.shosts
This file is used in exactly the same way as .rhosts. The purpose for having this file is to be able to use rhosts authentication with **ssh** without permitting login with **rlogin** or **rsh**.

/etc/hosts.equiv
This file is used during .rhosts authentication. It contains canonical host names, one per line. For more information on the format, see "sshd — OpenSSH daemon" on page 45. If the client host is found in this file, login is automatically permitted provided client and server user names are the same. Additionally, successful RSA host authentication is normally required. This file should only be writable by a superuser.

/etc/ssh/shosts.equiv
This file is processed exactly as **/etc/hosts.equiv**. This file may be useful to permit logins using **ssh**, but not using **rlogin** or **rsh**.

/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, see "sshd — OpenSSH daemon" on page 45.

$HOME/.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, "sshd — OpenSSH daemon" on page 45.

$HOME/.ssh/environment
Contains additional definitions for environment variables. For more information, see "Environment variables set by ssh" on page 29.

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. 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.

OpenSSH does not run in multibyte locales.

The SSH client cannot be run from OMVS. **ssh** has been disabled under OMVS because passwords are visible while they are being typed by the user in some situations.

Configuration files

ssh may additionally obtain configuration data from a per-user configuration file and a system-wide configuration file. For file format and configuration options, see “ssh_config – OpenSSH client configuration files” on page 56.

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**

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 [**–lLdDxX**] [**–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, it adds the files **$HOME/.ssh/id_rsa**, **$HOME/.ssh/id_dsa**, and **$HOME/.ssh/identity**. Alternative file names can be given on the command line. Where multiple MVS identities are mapped to the same UNIX UID, the home directory obtained by **ssh-add** for the current user is indeterminate and may not match the user’s $HOME variable. Users sharing a UNIX UID should always run **ssh-add** with arguments to specify the identities to be added. 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.

The authentication agent must be running and must be an ancestor of the current process for **ssh-add** to work.

Options

–d Instead of adding the identity, removes the identity from the agent. When run without specifying an identity to remove, it removes **$HOME/.ssh/id_rsa**, **$HOME/.ssh/id_dsa**, and **$HOME/.ssh/identity**. If the default identities are not present, **ssh-add** ends with return code 1.

Note: 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 smartcard 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 smartcard 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**.

–x Locks the agent with a password.

–X Unlocks the agent.

Files

$HOME/.ssh/identity
Contains the protocol version 1 RSA authentication identity of the user.

$HOME/.ssh/id_dsa
Contains the protocol version 2 DSA authentication identity of the user.

$HOME/.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

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 [**–a** *bind_address*] [**–c** | **–s**] [**–d**] [*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 **$HOME/.ssh/id_rsa**, **$HOME/.ssh/id_dsa**, and **$HOME/.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.

The idea 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 get an agent setup. 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. For example:

1. `ssh-agent $SHELL`
2. `eval 'ssh-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.

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 agents 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.

If a *command_string* is given, this is executed as a subprocess of the agent. When the command ends, so does the agent.

Environment variables

SHELL
: Contains the full path name of the current shell.

SSH_AGENT_PID
: Holds the agent's process ID.

SSH_AUTH_SOCK
: Holds the name of the socket through which the agent is accessible.

Files

$HOME/.ssh/identity
: Contains the protocol version 1 RSA authentication identity of the user.

$HOME/.ssh/id_dsa
: Contains the protocol version 2 DSA authentication identity of the user.

$HOME/.ssh/id_rsa
: Contains the protocol version 2 RSA authentication identity of the user.

/tmp/ssh-XXXXXXXX/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. “XXXXXXXX” will match ppid if the ppid is eight characters. Otherwise, “XXXXXXXX” is a system-generated string. These sockets should be readable only by the owner. The sockets should be automatically removed when the agent exits.

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/.

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* | **–d** [**–P** *passphrase*] [**–N** *new_passphrase*] [**–C** *comment*] [**–f** *output_keyfile*]

ssh-keygen –p [**–P** *old_passphrase*] [**–N** *new_passphrase*] [**–f** *keyfile*]

ssh-keygen –i | –X [**–f** *input_keyfile*]

ssh-keygen –e | –x [**–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 –D *reader*

ssh-keygen –U *reader* [**–f** *input_keyfile*]

Description

ssh-keygen generates, manages, and converts authentication keys for **ssh**. **ssh-keygen** 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 **–t** option.

Normally, each user wishing to use SSH with RSA or DSA authentication runs this once to create the authentication key in **$HOME/.ssh/identity**, **$HOME/.ssh/id_dsa**, or **$HOME/.ssh/id_rsa**. Where multiple MVS identities are mapped to the same UNIX UID, the home directory obtained by **ssh-keygen** for the current user is indeterminate and may not match the user's **$HOME** variable. Users with shared UNIX UIDs should specify the location of the key file being created with the **-f** option or override the default location prompt from **ssh-keygen** if it is incorrect. Additionally, the system administrator may use this to generate host keys.

Normally, 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. The passphrase may be empty to indicate no passphrase (host keys must have an empty passphrase) or it may be a string of arbitrary length. A series of words, punctuation, numbers, whitespace, or any string of characters. Unless it is empty, the passphrase must be greater than 4 characters long. However, good passphrases are 10-30 characters long, are not simple sentences or otherwise guessable (English prose has only 1-2 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.

There is no way to 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. After a key is generated, instructions below describe where the keys should be placed to be generated.

Options

–b *bits*
: Specifies the number of bits in the key to create. The minimum is 512 bits and the maximum is 32768. Generally, 1024 bits is considered sufficient and key sizes above that no longer improve security, but decrease performance. The default is 1024 bits. DSA key sizes are rounded off to the nearest multiple of 64 bits.

–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 **–C** option.

–C *comment*
: Provides the new comment. The comment is truncated after 1023 characters.

–d
: Specifies to create the dsa type key. Same as **–t** *dsa* option.

–D *reader*
: Not supported in z/OS UNIX. Downloads the RSA public key stored in the smartcard in reader.

–e
: Reads a private or public OpenSSH key file and prints the key in a 'SECSH Public Key File Format' to stdout. This option allows exporting keys for use

by several commercial SSH implementations and only applies to SSH protocol version 2. For more information on 'SECSH Public Key File Format', see J. Galbraith and R. Thayer, SECSH Public Key File Format which is a work in progress at the Internet Engineering Task Force Internet Drafts Index Web site.

–f *filename*
: Specifies the filename of the key file. The filename is truncated at 1023 characters including the 4 characters for ″.pub″ for the public keys.

–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 'SECSH Public Key File Format'. This option allows importing keys from several commercial SSH implementations. For more information on 'SECSH Public Key File Format', see J. Galbraith and R. Thayer, SECSH Public Key File Format which is a work in progress at the Internet Engineering Task Force Internet Drafts Index Web site.

–l
: Shows fingerprint of specified public key file. Private protocol version 1 RSA1 keys are also supported. For RSA nd 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`

–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.

  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 **–t** type or **–d** options are used, the **–P** value is used for passphrase regardless if **–N** is specified.

  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 script.

–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 filename to contain the private keys and passphrase, if **–P** or **–N**, and **–f** is not specified.

–U *reader*
: Not supported in z/OS UNIX; the command will fail. Uploads an existing RSA private key into the smartcard in reader.

–x
: Same as **–e**.

–X
: Same as **–i**.

–y Reads a private OpenSSH format file and prints an OpenSSH public key to stdout.

Exit values

0	Successful completion
> 0	Failure

Files

$HOME/.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.

$HOME/.ssh/identity.pub
Contains the protocol version 1 RSA public key for authentication. The contents of this file should be added to **$HOME/.ssh/authorized_keys** on all machines where the user wishes to log in using RSA authentication. There is no need to keep the contents of this file secret.

$HOME/.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.

$HOME/.ssh/id_dsa.pub
Contains the protocol version 2 DSA public key for authentication. The contents of this file should be added to **$HOME/.ssh/authorized_keys** on all machines where the user wishes to log in using RSA authentication. There is no need to keep the contents of this file secret.

$HOME/.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.

$HOME/.ssh/id_rsa.pub
Contains the protocol version 2 RSA public key for authentication. The contents of this file should be added to **$HOME/.ssh/authorized_keys** on all machines where the user wishes to log in using RSA authentication. There is no need to keep the contents of this file secret.

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 [**–v46**] [**–p** *port*] [**–T** *timeout*] [**–t** *type*] [**–f** *file*] [*host* | *addrlist namelist*] [...]

Description

ssh-keyscan is a utility for gathering the public **ssh** 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 not running **sshd** the public key information cannot be collected for that machine. The return value is not altered but a warning is displayed.

Example:

```
ssh_keyscan hostname1 hostname2
hostname1: exception!
(hostname2's rsa1 key displayed here)
```

Options

–f *filename*
: Reads hosts or *addrlist namelist* pairs from this file, one per line. If **–** is supplied instead of a file name, **ssh-keyscan** will read hosts or *addrlist namelist* pairs from the standard input.

–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.

–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.

File formats

Input format

```
1.2.3.4,1.2.4.4 name.my.domain,name,n.my.domain,n,1.2.3.4,1.2.4.4
```

Output format for rsa1 keys
host-or-namelist bits exponent modulus

Output format for rsa and dsa keys
host-or-namelist keytype bas64-encoded-key

Where keytype is either *ssh-rsa* **for an rsa key or** *ssh-dss* **for a dsa key**
/etc/ssh/ssh_known_hosts

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 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 — OpenSSH client (remote login program)” on page 22 and “sshd — OpenSSH daemon” on page 45 for more information about host-based authentication.

ssh-keysign is disabled by default and can only be enabled in the global client configuration file **/etc/ssh/ssh_config** by setting *HostbasedAuthentication* to "yes".

Files

/etc/ssh/ssh_config
: Controls whether **ssh-keysign** is enabled. HostbasedAuthentication 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. Because they are readable only by UID 0, **ssh-keysign** must be setuid 0 if host-based authentication is used.

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 [**–vxXh**] [**–b** *bytes*]

Description

ssh-rand-helper is a small helper program used by **ssh**, **ssh-add**, **ssh-agent**, **ssh-keygen**, **ssh-keyscan**, and **sshd** to gather random numbers of cryptographic quality.

Normally **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 system 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. **ssh-rand-helper** also stores seed files in **~/.ssh/prng_seed** between executions.

Options

This program is not intended to be run by the end 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 message. Multiple **–v** options increase the debugging level.
–x	Outputs a hexadecimal instead of a binary seed.
–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.

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 [**–deiqtD46**] [**–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 (SSH 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 is the daemon that listens for connections from clients. It is normally started when z/OS UNIX is initialized. (See Chapter 2, "For system administrators: Setting up and starting the sshd daemon," on page 3 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.

SSH protocol version 1

Each host has a host-specific RSA key (normally 1024 bits) used to identify the host. Additionally, when the daemon starts, it generates a server RSA key (normally 768 bits). This key is normally regenerated every hour if it has been used and it 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 the default. The client selects the encryption algorithm to use from those offered by the server.

Next, the server and client enter an authentication dialog. The client tries to authenticate itself using .rhosts authentication, .rhosts authentication combined with RSA host authentication, RSA challenge-response authentication, or password based authentication.

Rhosts authentication is normally disabled because it is fundamentally insecure, but can be enabled in the server configuration file if desired. System security is not improved unless **rshd**, **rlogind**, and **rexecd** are disabled (thus completely disabling **rlogin** and **rsh** into the machine).

SSH protocol version 2

Version 2 works similarly to version 1; each host has a host-specific key (RSA or DSA) used to identify the host. However, when the daemon starts, it does not generate a server key. 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, currently 128-bit AES, Blowfish, 3DES, CAST128, Arcfour, 192-bit AES, or 256-bit AES. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (hmac-sha1 or hmac-md5).

Protocol version 2 provides a public key based user (PubkeyAuthentication) or client host (HostbasedAuthentication) authentication method, conventional password authentication and challenge response based methods.

Command execution and data forwarding

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time, the client may request things like allocating a pseudo-tty, forwarding X11 connections, forwarding TCP/IP connections, or forwarding the authentication agent connection over the secure channel.

Finally, the client either requests a shell or execution of a command. The sides then enter session mode. In this mode, either side may 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.

sshd can be configured using command-line options or a configuration file. Command-line options override values specified in the configuration file.

sshd rereads its configuration file when it receives a hangup signal, SIGHUP, by executing itself with the name it was started as, such as **/usr/sbin/sshd**.

Options

–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. Maximum is 3.

–D
: **sshd** does not fork and does not become a daemon. This allows for easy monitoring of **sshd**.

–e
: **sshd** sends the output to the 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 *configuration_file*
: Specifies the name of the configuration file. The default is **/etc/ssh/sshd_config**. **sshd** will not start if there is no 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. This option must be given if **sshd** is not run as UID(0) (as the normal host key files are normally not readable by anyone but superuser). The default is **/etc/ssh/ssh_host_key** for protocol version 1 and **/etc/ssh/ssh_host_rsa_key** and **/etc/ssh/ssh_host_dsa_key** for protocol version 2. It is possible to have multiple host key files 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 may 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** may 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 configuration file. This is useful for specifying options for which there is no separate command-line flag.

–p *port*
: Specifies the port on which the server listens for connections (default 22).

Multiple port options are permitted. Ports specified in the configuration file are ignored when a command-line port is specified.

–q Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged.

–t Test mode. Only check the validity of the configuration file and sanity of the keys. This is useful for updating **sshd** reliably as configuration options may 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** is also used to prevent **sshd** from making DNS requests unless the authentication mechanism or configuration requires it. Authentication mechanisms that may 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.

–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.

Login process

When a user successfully logs in, **sshd** does the following:

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 **$HOME/.hushlogin**; see “Files” on page 51 for details).
2. If the login is on a tty, records login time to utmpx database.
3. If the user is not a superuser, checks **/etc/nologin**; if it exists, prints contents and quits.
4. Changes to run with normal user privileges.
5. Sets up basic environment.
6. Reads **$HOME/.ssh/environment** if it exists and users are allowed to change their environment. See the **PermitUserEnvironment** option in “ssh_config – OpenSSH client configuration files” on page 56.
7. Changes to user’s home directory.
8. If **$HOME/.ssh/rc** exists, runs it; or, if **/etc/ssh/sshrc** exists, runs it; otherwise runs **xauth**. 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 user’s shell or command.

Authorized_keys file format

$HOME/.ssh/authorized_keys is the default file that lists the public keys that are permitted for RSA authentication in protocol version 1 and for public key authentication (PubkeyAuthentication) in protocol version 2. AuthorizedKeysFile may be used to specify an alternate file.

Each line of the file contains one key (empty lines and lines starting with # are ignored as comments). Each RSA public key consists of the following fields, separated by spaces: options, bits, exponent, modulus, comment. Each protocol version 2 public key consists of: options, key-type, base64 encoded key, comment. The options field is optional; its presence is determined by whether the line starts with a number or not (the options field never starts with a number). The bits, exponent, modulus, and comment fields give the RSA key for protocol version 1. For protocol version 2, the keytype is ″ssh-dss″ or ″ssh-rsa″.

Lines in this file are usually several hundred bytes long (because of the size of the public key modulus). To avoid typing them in, 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 1 and protocol 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 case-insensitive):

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 (″`*`″ and ″`?`″ serve as wildcards). The list may also contain patterns negated by prefixing them with ″`!`″; if the canonical host name matches a negated pattern, the key is not accepted. 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 (name servers and routers would have to be compromised in addition to just the key).

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 pty if the client requests a pty; otherwise it is run without a tty. If an 8-bit clean channel is required, one must not request a pty or should specify no-pty. A quote may 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 may specify any combination of TCP/IP and X11 forwarding unless they are explicitly prohibited. 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 set by ssh” on page 29 in “ssh — OpenSSH client (remote login program)” on page 22 for more information. Multiple options of this type are permitted. Environment processing is disabled by default and is controlled via the *PermitUserEnvironment* option. This option is automatically disabled if UseLogin is enabled.

no-agent-forwarding
: Forbids authentication agent forwarding when this key is used for authentication.

no-port-forwarding
: Forbids TCP/IP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This can be used in conjunction with the command option.

no-pty
: Prevents tty allocation (a request to allocate a pty will fail).

no-X11-forwarding
: Forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

permitopen="host:port"
: Limit local **ssh –L** port forwarding such that it may only connect to the specified host and port. IPv6 addresses can be specified with an alternate syntax: *host*/*port*. Multiple *permitopen* options may be separated by commas. No pattern matching is performed on the specified hostnames.

Example of format:

```
1024 33 12121...312314325 ylo@foo.bar

from="*.niksula.hut.fi,!pc.niksula.hut.fi" 1024 35 23...2334 ylo@niksula

command="dump /home",no-pty,no-port-forwarding 1024 33 23...2323 backup.hut.fi

permitopen="10.2.1.55:80",permitopen="10.2.1.56:25" 1024 33 23...2323
```

SSH_KNOWN_HOSTS file format

The **/etc/ssh/ssh_known_hosts**, and **$HOME/.ssh/known_hosts** files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional) and the per-user file is maintained automatically. Whenever the user connects from an unknown host, its key is added to the per-user file.

Each line in these files contains the following fields: hostnames, bits, exponent, modulus, comment. The fields are separated by spaces.

Hostnames is a comma-separated list of patterns (`*` and `?` act as wild cards). 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 may 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.

Bits, exponent, and modulus are taken directly from the RSA host key. They can generally be obtained from **/etc/ssh/ssh_host_key.pub**. 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.

Example of format :

```
closenet,...,130.223.208.41 1024 37 159...93 closenet.hut.fi
cvs.openbsd.org,199.185.137.3 ssh-rsa AAAA1234.....=
```

Files

/etc/ssh/sshd_config
: Contains configuration data for **sshd**. The file format and configuration options are described in **sshd_config**.

/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 should only be owned and readable by a superuser. **sshd** does not start if this file is 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 should be world-readable, but writable only by a superuser. Their contents should match the respective private parts. 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**.

/etc/ssh/moduli
: Contains Diffie-Hellman groups used for the ″Diffie-Hellman Group Exchange″. The file format is described in **moduli**.

/var/empty
: **chroot** directory used by **sshd** during privilege separation in the pre-authentication phase. The directory should not contain any files and must be owned by a superuser and not be group-writable or world-writable.

/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 content of this file is not sensitive. It can be world-readable. This file is not created if the server is running in debug mode.

$HOME/.ssh/authorized_keys
: Lists the public keys (RSA or DSA) that can be used to log into the user's account. This file must be readable by a superuser (which may on some machines be implicitly world-readable if the user's home directory resides on an NFS volume). It is recommended that it not be accessible by others. For file format information see "Authorized_keys file format" on page 48. Users will place the contents of one or more of their identity.pub, id_dsa.pub, and id_rsa.pub files into this file, as described in "ssh-keygen — Authentication key generation, management, and conversion" on page 38.

/etc/ssh/ssh_known_hosts, $HOME/.ssh/known_hosts
: These files are consulted when using rhosts with RSA host authentication or protocol version 2 host-based authentication to check the public key of the host. The key must be listed in one of these files to be accepted. The client uses the same files to verify that it is connecting to the correct remote host. These files should be writable only by a superuser or the owner. **/etc/ssh/ssh_known_hosts** should be world-readable and **$HOME/.ssh/known_hosts** can, but need not be world-readable.

/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 should be world-readable.

/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.

$HOME/.rhosts

This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without password. On many historical UNIX platforms the same file is used by **rlogind** and **rshd**. The file must be writable only by the user. It is recommended that it not be accessible by others.

$HOME/.shosts
For **ssh** this file is exactly the same as for .rhosts. However, this file is not used by **rlogin** and **rshd**, so using this permits access using SSH only.

/etc/hosts.equiv

This file is used during .rhosts 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 may 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 RSA host authentication is normally required. This file should be writable only by a superuser. It is recommended that it be world-readable.

Guideline: Do not use user names in **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/ssh/shosts.equiv
This is processed exactly as **/etc/hosts.equiv**. However, this file can be useful in environments that want to run both **rsh**, **rlogin**, and **ssh**.

$HOME/.ssh/environment
This file is read into the environment at login (if it exists). It can only contain empty lines, comment lines (starting with #), and assignment lines of the form *name=value*. The file should be writable only by the user; it need not be readable by anyone else. Environment processing is disabled by default and is controlled via the *PermitUserEnvironment* option.

$HOME/.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 10.

The primary purpose of this file is to run any initialization routines which may 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/ssh/sshrc
Like **$HOME/.ssh/rc**. This can be used to specify machine-specific login-time initialization globally. This file should be writable only by superuser and world-readable.

$HOME/.hushlogin
If this file exists, the message of the day and last login time are not displayed.

Configuration files

sshd reads configuration data from **/etc/ssh/sshd_config** (or the file specified with **–f** on the command line). For file format and configuration options, see "ssh_config – OpenSSH client configuration files" on page 56.

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.

Restriction: OpenSSH does not run in multibyte locales.

Limitations

The maximum length of the ephemeral server key is INT_MAX.

Related information

moduli, **scp**, **sftp**, **sftp-server**, **ssh**, **ssh-add**, **ssh-agent**, **ssh-keygen**, **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 5. 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

ssh_config – OpenSSH client configuration files

Description

ssh obtains configuration data from the following sources in the following order:

1. Command line options
2. User's configuration file (**$HOME/.ssh/config**)
3. System-wide configuration file (**/etc/ssh/ssh_config**)

For each parameter, the first obtained value is used. The configuration files contain sections bracketed 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.

Since the first obtained value for each parameter is used, more host-specific declarations should be given near the beginning of the file, and general defaults at the end.

Format

The configuration file views empty lines and lines starting with # as comments. Otherwise, a line is of the format *keyword arguments*. Configuration options may 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.

The latter format is useful to avoid the need to quote white space when specifying configuration options using the **scp**, **sftp –o** and **ssh** options.

Example:

```
keyword argument
keyword=argument
```

Keywords are case-insensitive and arguments are case-sensitive. The following are the possible keywords:

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". This 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" (default). An SSH agent, Kerberos authentication (if available), or trusted host authentication must be used for authentication to succeed in batch mode.

BindAddress
Specifies the interface to transmit from on machines with multiple interfaces or aliased addresses. This option 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" (default) or "no".

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 hostname. 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 $HOME/.ssh/known_hosts. 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*, *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. The default is:

```
"aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc"
```

Valid ciphers include :

3des-cbc
Triple DES algorithm (3DES)

blowfish-cbc
Blowfish algorithm

cast128-cbc
CAST algorithm

arcfour
ARCFOUR algorithm

aes128-cbc
Advanced Encryption Standard (AES) with 128-bit key

aes192-cbc
Advanced Encryption Standard (AES) with 192-bit key

aes256-cbc
Advanced Encryption Standard (AES) with 256-bit key

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" (default).

Compression
Specifies whether to use compression. The argument must be set to "yes" or "no" (default).

CompressionLevel
For protocol version 1, 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 appropriate for most applications.

ConnectionAttempts
Specifies the number of tries (one per second) to make before exiting. The argument must be an integer. This may be useful in scripts if the connection sometimes fails. The default is 1.

DynamicForward
Specifies that a TCP/IP port on the local machine be forwarded over 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. Currently, the SOCKS4 protocol is supported and **ssh** acts as a SOCKS4 server. Multiple forwardings may be specified and additional forwarding can be given on the command line. Only the superuser can forward privileged ports.

EscapeChar
Sets the escape character (default of ~). The escape character can also be set on the command line. The argument should 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" (default).

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" (default).

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. An attacker may then be able to perform activities such as keystroke monitoring.

GatewayPorts
Specifies whether remote hosts are allowed to connect to local forwarded ports. By default, **ssh** binds local port forwardings to the loopback address. This prevents other remote hosts from connecting to forwarded ports. GatewayPorts can be used to specify that **ssh** should 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" (default).

GlobalKnownHostsFile
Specifies a file to use for the global host key database instead of **/etc/ssh/ssh_known_hosts**.

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. * and ? can be used as wildcards in the patterns. 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).

HostbasedAuthentication
Specifies whether to try rhosts based authentication with public key authentication. The argument must be set to "yes" or "no" (default). This option applies to protocol version 2 only and is similar to RhostsRSAAuthentication.

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. This can be used 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.

IdentityFile
Specifies a file from which the user's RSA or DSA authentication identity is read. The default is **$HOME/.ssh/identity** for protocol version 1 and **$HOME/.ssh/id_rsa** and **$HOME/.ssh/id_dsa** for protocol version 2. Additionally, any identities represented by the authentication agent will be used for authentication. The file name may use the tilde syntax to refer to a user's home directory. It is possible to have multiple identity files specified in configuration files; all these identities will be tried in sequence.

KeepAlive
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".

KerberosAuthentication
: Not supported on z/OS UNIX. Specifies whether Kerberos authentication will be used. The argument must be set to "yes" or "no".

KerberosTgtPassing
: Not supported on z/OS UNIX. Specifies whether a Kerberos TGT will be forwarded to the server. This will work only if the Kerberos server is actually an AFS kaserver. The argument must be set to "yes" or "no".

LocalForward
: Specifies that a TCP/IP port on the local machine be forwarded over the secure channel to the specified host and port from the remote machine. The first argument must be a port number, and the second must be *host*:*port*. IPv6 addresses can be specified with an alternate syntax: *host*/*port*. Multiple forwardings may be specified and additional forwardings can be given on the command line. Only the superuser can forward privileged ports.

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 in protocol version 2 for data integrity protection. Multiple algorithms must be comma-separated. The default is *hmac-md5,hmac-sha1,hmac-ripemd160,hmac-sha1-96,hmac-md5-96*.

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" (default, to check the host key for localhost).

NumberOfPasswordPrompts
: Specifies the number of password prompts before giving up. The argument must be an integer. Default is 3.

Note: 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".

Port Specifies the port number to connect to on the remote host. 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**. Please note that **keyboard-interactive** is 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 **/bin/sh**. 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). *CheckHostIP* 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".

RemoteForward
Specifies that a TCP/IP port on the remote machine be forwarded over the secure channel to the specified host and port from the local machine. The first argument must be a port number and the second must be *host*:*port*. IPv6 addresses can be specified with an alternate syntax: *host*/*port*. Multiple forwardings may be specified and additional forwardings can be given on the command line. 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 has no affect on security. Most servers do not permit RhostsAuthentication because it is not secure. The argument must be set to "yes" or "no" (default). This option requires **ssh** to be setuid 0 and *UsePrivilegedPort* to be set to "yes".

Note: *RhostsAuthentication* cannot be used with privilege separation. For more information on privilege separation, see "sshd — OpenSSH daemon" on page 45.

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" (default).

RSAAuthentication
This option applies to protocol version 1 only. 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.

SmartcardDevice
Not supported on z/OS UNIX. Specifies which smartcard device to use. The

argument to this keyword is the device that **ssh** should use to communicate with a smartcard used for storing the user's private RSA key. By default, no device is specified and smartcard support is not activated.

StrictHostKeyChecking
: If the argument is set to "yes", **ssh** will never automatically add host keys to the **$HOME/.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"(default).

UsePrivilegedPort
: Specifies whether to use a privileged port for outgoing connections. This option must be set to "yes" if RhostsAuthentication and RhostsRSAAuthentication authentications are needed with servers which only support protocol version 1. The argument must be set to "yes" or "no" (default). If 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. This saves the trouble of having 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 **$HOME/.ssh/known_hosts**.

XAuthLocation
: Specifies the full pathname 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 10.

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.

Files

$HOME/.ssh/config
: This is the per-user configuration file. For the format of this file see "Format" on page 56. The file is used by the **ssh** client. This file does not usually contain any sensitive information, but the recommended permissions are read/write for the user and not accessible by others.

/etc/ssh/ssh_config
: 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

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.

sshd_config – OpenSSH daemon configuration files

Format

/etc/ssh/sshd_config

Description

sshd reads configuration data from **/etc/ssh/sshd_config** or the file specified with **-f** on the command line). “File format” describes the file format.

File format

The configuration file views empty lines and lines starting with # as comments. Otherwise, a line is of the format *keyword arguments*. Keywords are case-insensitive and arguments are case-sensitive. The following are the possible keywords:

AFSTokenPassing
Not supported on z/OS UNIX. Specifies whether an AFS token may 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. * and ? can be used as wild cards in the patterns. Only group names are valid; a numerical group ID is not recognized. By default, login is allowed for all groups.

AllowTcpForwarding
Specifies whether TCP forwarding is permitted. Disabling TCP forwarding does not improve security unless users are also denied shell access, as they can install their own forwarders. The default is "yes".

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. * and ? can be used as wild cards in 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.

AuthorizedKeysFile
Specifies the file that contains the public keys that can be used for user authentication. AuthorizedKeysFile may 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 given). The default is *.ssh/authorized_keys* anchored off the user's home directory.

Banner
In some jurisdictions, sending a warning message before authentication may be relevant for obtaining legal protection. The contents of the specified file are sent to the remote user before authentication is allowed. This option is only available for protocol version 2. The default is no banner is displayed.

ChallengeResponseAuthentication
Not supported on z/OS UNIX. Specifies whether challenge response authentication is allowed. The default is "yes".

Ciphers
Specifies the ciphers to use for encrypting the session in protocol version 2. Multiple ciphers must be comma-separated. The default is:

```
"aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc"
```

Valid ciphers include :

3des-cbc
a Triple-DES (3DES) algorithm

blowfish-cbc
Blowfish algorithm

cast128-cbc
CAST algorithm

arcfour
ARCFOUR algorithm

aes128-cbc
Advanced Encryption Standard (AES) with 128-bit key

aes192-cbc
Advanced Encryption Standard (AES) with 192-bit key

aes256-cbc
Advanced Encryption Standard (AES) with 256-bit key

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. This option applies to protocol version 2 only. The default is 0, indicating that these messages will not be sent to the client.

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 *KeepAlive* Because the client alive messages are sent through the encrypted channel, they are not spoofable. The TCP keepalive option enabled by *KeepAlive* 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.

Compression
Specifies whether compression is allowed. The argument must be set to "no" (default) or "yes". On z/OS, compression cannot be used with privilege separation.

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. * and ? can be used as wildcards in the patterns. Only group names are valid; a numerical group ID is not recognized. The default is to allow login for all groups.

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. * and ? can be used as wild cards in 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.

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** should bind remote port forwardings to the wildcard address, thus allowing remote hosts to connect to forwarded ports. The argument must be set to "yes" or "no" (default).

HostbasedAuthentication
Specifies whether rhosts or **/etc/hosts/.equiv** authentication together with successful public key client host authentication is allowed (host-based authentication). This option applies to protocol version 2 only and is similar to RhostsRSAAuthentication. The default is "no".

HostKey
Specifies a file containing a private host key used by SSH. The default is **/etc/ssh/ssh_host_key** for protocol version 1 and **/etc/ssh/ssh_host_rsa_key** and **/etc/ssh/ssh_host_dsa_key** for protocol version 2. **sshd** will refuse to use a file if it is group/world-accessible. It is possible to have multiple host key files. *rsa1* keys are used for protocol version 1 and *dsa* or *rsa* are used for protocol version 2.

IgnoreRhosts
Specifies that .rhosts and .shosts files will not be used in RhostsAuthentication, RhostsRSAAuthentication or HostbasedAuthentication.

/etc/hosts.equiv and **/etc/ssh/shosts.equiv** are still used. The default is "yes".

IgnoreUserKnownHosts
Specifies whether **sshd** should ignore the user's **$HOME/.ssh/known_hosts** during RhostsRSAAuthentication or HostbasedAuthentication. The default is "no".

KeepAlive
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, 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 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. This 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".

KerberosOrLocalPasswd
Not supported on z/OS UNIX. If set, then the password will be validated via 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 may 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 destroy 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). The purpose of regeneration is to prevent decrypting 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 may be used:

```
ListenAddress host|IPv4addr|IPv6_addr
ListenAddress host|IPv4_addr:port
ListenAddress[host|IPv6_addr]:port
```

If port is not specified, **sshd** will listen 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. Logging with a DEBUG level violates the privacy of users and is not recommended.

MACs Specifies the available MAC (message authentication code) algorithms. The MAC algorithm is used in protocol version 2 for data integrity protection. Multiple algorithms must be comma-separated. The default is "hmac-md5,hmac-sha1,hmac-ripemd160,hmac-sha1-96,hmac-md5-96".

MaxStartups
Specifies the maximum number of concurrent unauthenticated connections to the **sshd** 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).

PAMAuthenticationVaKbdInt
Not supported on z/OS UNIX. Specifies whether PAM challenge response authentication is allowed. This 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 default is "yes".

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 may 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 userid *uuuuuuuu* without a password.

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.

PermitUserEnvironment
Specifies whether **~/.ssh/environment** and **environment=** options in **~/.ssh/authorized_keys** are processed by **sshd**. The default is "no". Enabling environment processing may enable users to bypass access restrictions in some configurations using mechanisms such as LD_PRELOAD.

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
Specifies whether **sshd** should print the date and time when the user last logged in. The default is "yes". 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".

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". 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". This option apples to protocol version 1 only. RhostsAuthentication cannot be used with privilege separation.

RhostsRSAAuthentication
Specifies whether rhosts or **/etc/hosts.equiv** authentication together with successful RSA host authentication is allowed. The default is "no". This option applies to protocol version 1 only.

RSAAuthentication
Specifies whether pure RSA authentication is allowed. The default is "yes". 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 the following files and directories are owned by the current user or superuser and that they are not group or world-readable:

- 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 to execute upon subsystem request. The command **sftp-server** implements the **sftp** file transfer subsystem. By default, no subsystems are defined. User-defined (non-builtin) subsystems are only supported between z/OS and z/OS. See “Limitations” on page 70 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.

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, it will be disabled after authentication. 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
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 security in any way, as 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** may 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. 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 may 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

Chapter 6. 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 the 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 specifies is openssh format.

FOTS0106 input line too long

Explanation: ssh-keygen could not convert the key. Data in the keyfile had a too long line.

System Action: Command ends.

User Response: Check that you specified the correct keyfile 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 keyfile 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 keyfile 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/change keyfile, check that there is enough space to create a keyfile.

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 keyfile 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 keyfile 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 .ssh 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: A prompt to specify the 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 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 512 to 32768

System Action: Command ends.

User Response: Change the bits value and reissue 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 keyfile. 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 keyfile 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 keyfile and verify that 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 keyfile had a line that was too long.

System Action: Command ends.

User Response: Check that you specified the correct keyfile 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 keyfile 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 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.

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 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 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 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 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 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 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 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 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 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 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 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.

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:

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 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.***'.**

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 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 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: fnctl() system call failed.

System Action: Command ends

User Response: Refer to *z/OS 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 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 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 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 is 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.

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.

FOTS0505 *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 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 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 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 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 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 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 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 *msg_len* **< 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.

FOTS0801 pipe: *system error*

Explanation: System call pipe() failed.

System Action: Command ends.

User Response: Refer to *z/OS 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 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 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 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 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 *pathname*.

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 missed 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 missed 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 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 Invalid flag -*flag*

Explanation: You specified an invalid flag after interactive command *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 interactive command.

System Action: Command continues.

User Response: Check *IBM Ported Tools for z/OS User's Guide* for a list of valid interactive commands.

FOTS0823 You must specify at least one path after a *command* **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 interactive command.

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 interactive command.

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 *command* 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 directory because the argument specified after interactive command *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 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 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(s) 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 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 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 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 interactive command 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 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 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 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 type*) packet, got *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.

FOTS0846 Expected SSH2_FXP_HANDLE(*handle*) packet, got *handle*

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 type*) packet, got *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.

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.

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:

System Programmer Response:

FOTS0859 Expected SSH2_FXP_NAME(*packet type***) packet, got** *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.

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 interactive command *ln* and *symlink*.

System Action: The program continues.

User Response: Do not use interactive command *symlink* or *ln* .

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(s).

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 type***) packet, got** *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.

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.

FOTS0857 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 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 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 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 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 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 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 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 type***) packet, got** *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.

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 *"remote file"***:** *error message*

Explanation: sftp failed to write to the remote file (while attempting to upload file) due to the displayed error message.

System Action: The program ends.

User Response: 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 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.

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.

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 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 not supported 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 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 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: You entered passphrase is less than 4 bytes which is not allowed by ssh-keygen.

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:

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 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:

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 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: 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: cannot handle BN of size** *bytes*

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.

FOTS0944 buffer_get_bignum: input buffer too small

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.

FOTS0946 buffer_get_bignum2: cannot handle BN of size *bytes*

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.

FOTS0947 buffer_get_string: bad string length *number*

Explanation: Internal error. Received string too long.

System Action: The program ends.

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 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 programc 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: 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 ends.

User Response: Contact your system programmer.

System Programmer Response: Follow local procedures for reporting problems to IBM.

FOTS0952 buffer_consume: 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.

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: bad string length *number*

Explanation: Internal error. Received string too long.

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 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 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 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 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 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 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 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 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 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 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 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 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 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 *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 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.

FOTS0981 setsockopt IPV6_V6ONLY: *system error*

Explanation: A call to setsockopt() failed. IPV6_V6ONLY is the one of the arguments of setsockopt(). The system error is displayed.

System Action: The program continues.

User Response: Refer to *z/OS 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.

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 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 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 of the *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*

System Action: The program continues.

User Response: Refer to ssh of the *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.

System Action: The program continues.

User Response: Refer to ssh of the *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.

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 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.

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_request_remote_forwarding: too many forwards

Explanation: A request for forwarding an application over a new channel was denied because the internal maximum of 99 forwarded channels has been reached.

System Action: The program ends.

User Response: Check to make sure you do not request forwarding of more than 99 applications. 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 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 fucntion 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 fucntion 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 fucntion 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 fucntion 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 fucntion EVP_Cipher() for more information. If unable to resolve, contact your system programmer.

System Programmer Response: Follow local procedures for reporting problems to IBM. EVP_CIPHER_CT_ is defined in openssl crypto/evp/evp.h

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***: ad 3des iv 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.

FOTS1026 *function***: no 3des 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.

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.

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_num*

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.

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 ends.

User Response: Check your primes file to make sure the displayed *line_num* exist. 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 ends.

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_compute: 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.

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 ends.

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.

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.

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’ or ’d’. This message can also be displayed for an internal error.

System Action: The program ends.

User Response: Check to make sure you specify an valid key type after option ’t’ or ’d’. 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.

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 *proposal* **server** *proposal*

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 with 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 no kex alg

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 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 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: The call to BN_new 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.

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 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 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: *proc 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 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.

System Action: Command continues.

User Response: Refer to *z/OS 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.

System Action: Command continues.

User Response: Refer to *z/OS 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.

FOTS1237 Could not create directory *dirname***:** *Errno string*

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 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 request for local forwarding has failed.

System Action: The program continues.

User Response: Check the more descriptive error message displayed with this message.

System Programmer Response: None.

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 exits with an error.

User Response: Refer to*z/OS 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 (*max*) that can be specified in configuration files or the command line has been exceeded.

System Action: The program exits with an error.

User Response: Reissue the command with a smaller number of identity files.

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 exits with an error.

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 with an error.

User Response: Reissue ssh with a command or without the -f option.

FOTS1243 Can't open user config file ***filename*****:**

Explanation: ssh was unable to open the user configuration file *filename*. The system error is displayed.

System Action: The program exits with an error.

User Response: Refer to *z/OS 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 exits with an error.

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 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 exits with an error.

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 exits with an error.

User Response: Refer to *z/OS 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.

FOTS1252 The SSH client cannot be run under OMVS.

Explanation: The SSH client cannot be run under OMVS due to password visibility issues.

System Action: The program exits with an error.

User Response: Reissue the command from a non-OMVS environment.

System Programmer Response:

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 the filename specified is correct and exists.

FOTS1288 no support for smartcards.

Explanation: ssh on z/OS does not provide support for smartcards.

System Action: The program continues.

User Response: Reissue command without smartcard 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 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 exits.

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 exits.

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 exits.

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 exits.

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 exits.

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: The program exits.

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 exits.

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 exits.

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 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 exits.

User Response: Refer to *z/OS 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 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 *host*.

System Action: The program exits.

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.

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:

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 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.

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*:*lineno*

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 exits.

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 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 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 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 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 exits.

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 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 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 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 exits.

User Response: Refer to *z/OS 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: *system error*

Explanation: A call to fork() failed. The system error is displayed with this message.

System Action: The program exits.

User Response: Refer to *z/OS 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***:** *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 exits.

User Response: Refer to *z/OS 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 exits.

User Response: Verify the remote host is still operable. Verify 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 exits.

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 exits.

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 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 rsa1 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 respond_to_rsa_challenge: BN_new failed

Explanation: During key exchange, the ssh client could not obtain a session key. 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.

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 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 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 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.

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 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 the ssh-keysign and ssh client which are 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 the DNS setup on the local system. 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 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 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 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 exits.

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 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.

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 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 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 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 exits.

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 exits.

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 exits.

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 *max forwards*

System Action: The program exits.

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 exits.

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 exits.

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 exits.

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 exits.

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: 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 exits.

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.

FOTS1397 *filename* **line** *line number*: **Too many identity files specified (max** *max***).**

Explanation: The maximum number of authentication identity files (*max*) that can be specified in configuration files or command line has been exceeded.

System Action: The program exits with an error.

User Response: Reissue the command with a smaller number of identity files. Check the number of times the IdentityFile configuration option was specified in the configuration file.

System Programmer Response: Follow local procedures for reporting problems to IBM.

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.

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.

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 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 have the 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 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 have the 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 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 second argument.**

Explanation: While parsing *filename*, the second 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 have the 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.

FOTS1417 *filename* **line** *lineno*: **Bad forwarding specification.**

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 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 exits.

User Response: Check the specified line number in the file to make sure 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 either the *DynamicForward* configuration option.

System Action: The program ends.

User Response: Check the specified line number in the file to make sure 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 exits.

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 exits.

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 exits.

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 exits.

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 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 exits.

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 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 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 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 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 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 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 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 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 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_connection: 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 *username* **does not exist**

Explanation: If sshd is running with configuration option UsePrivilegeSeparation enabled, the user *username* must exist.

System Action: The program ends.

System Programmer Response: For more information on sshd setup, see *IBM Ported Tools for z/OS User's Guide*.

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 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 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 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 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 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.

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.

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 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.

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.

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 the *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 the *IBM Ported Tools for z/OS User's Guide* for more information on sshd -k.

FOTS1492 too many host keys.

Explanation: The number of host keys specified with sshd -h option is greater than the maximum sshd allows (currently 256).

System Action: The program ends.

System Programmer Response: Reissue sshd with a smaller number of host keys.

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 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.

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 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 would 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: An call to __tcsetcp() failed while sshd was trying to set the code page for the master pty. The system error is displayed with this message.

System Action: The program continues.

System Programmer Response: Refer to *z/OS 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 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: An 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: An 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: An 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: An 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 C/C++ Run-Time Library Referencez/OS 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 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 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 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* **A call to getpwuid() failed for** *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 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 in 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 in 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 exits.

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 ssh client 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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno***: missing inet addr.**

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** *lineno***: 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** *lineno***: bad inet addr: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** *lineno***: bad port number.**

Explanation:

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** *lineno***: 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** *lineno***: too many host keys specified (max** *hostkeys***).**

Explanation: The sshd HostKey option was specified more times than sshd supports. The maximum number of HostKey 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 HostKey option which caused this error. Reissue sshd with a valid number of HostKey options.

FOTS1595 *filename* **line** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno***: 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** *lineno*: **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** *lineno*: **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** *lineno***d: 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 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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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** *lineno*: **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.

FOTS1620 *filename* **line** *lineno*: **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** *lineno*: **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 *line number* of 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 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 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 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 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 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 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 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 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 the command specified for the subsystem (in the sshd configuration file) is in the search order specified by PATH. Refer to *z/OS 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 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 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 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 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 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 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 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 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.

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.

FOTS1671 Bad line *lineno* **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 the above line.

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 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 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: *addr2*(*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 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 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.

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: fcntl() system call failed.

System Action: The program ends.

User Response: Refer to *z/OS 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.

FOTS1740 *function***: socketpair**

Explanation: socketpair() system call failed.

System Action: The program ends.

User Response: Refer to *z/OS 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 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 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 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 ***cp0*****: 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 exists 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 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 key exists 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 hostkey found.

System Action: The program ends.

User Response: Verify the host key exists 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 *vect[0]***: set times:** *error_message*

Explanation: utimes() system call failed.

System Action: The program ends.

User Response: Refer to *z/OS 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 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 *np***: truncate:** *error_messages*

Explanation: ftruncate() system call failed.

System Action: The program ends.

User Response: Refer to *z/OS 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 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 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.

FOTS1801 Couldn't create socket: *error_message*

Explanation: socket() system call failed.

System Action: The program continues.

User Response: Refer to *z/OS 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 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 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 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 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 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 the sure ssh_prng_cmds file is setup 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 the sure ssh_prng_cmds file is setup 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 the sure ssh_prng_cmds file is setup 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 the sure ssh_prng_cmds file is setup 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 the sure ssh_prng_cmds file is setup properly. See the ssh-rand-helper man page for information.

FOTS1812 ignored extra commands (max *maximum***),** *filename 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 the sure ssh_prng_cmds file is setup 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 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 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 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 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 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 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.

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 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 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 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.

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.

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_code*

Explanation: fnctl() system call failed.

System Action: The program continues.

User Response: Refer to *z/OS 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.

FOTS1921 setsockopt IPTOS_LOWDELAY: *error_code*

Explanation: setsockopt() system call failed.

System Action: The program continues.

User Response: Refer to *z/OS 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 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 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 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.

FOTS1941 Couldn't open /dev/null: *error_message*

Explanation: open() system call failed.

System Action: The program ends.

User Response: Refer to *z/OS 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.

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 *z/OS 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 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 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 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 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 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 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 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 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 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 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 continues.

User Response: Contact your system programmer to report the problem.

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 *slen* **slen2** *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.

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 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 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 1024 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 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 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.

FOTS2037 setgroups: *error_message*

Explanation: setgroups() system call failed.

System Action: The program ends.

User Response: Refer to *z/OS 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 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 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.

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 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: setuid() system call failed.

System Action: The program ends.

User Response: Refer to *z/OS 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 allocated 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 allocated 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 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.

FOTS2057 *fund***: 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 *fund***: 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 *fund***: 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.

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 init mac** *mac_name*

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.

FOTS2068 *fund***: 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.

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.

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.

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 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 ends.

User Response: Refer to *z/OS 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 ends.

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 ends.

User Response: Contact your system programmer to report the problem.

System Programmer Response: Follow local procedures for reporting problems to IBM.

Appendix A. OpenSSH files Quick Reference

Configuration files

Samples provided by the installation must be copied into **/etc**.

File	Copied to	Description	Permissions	Owner
/samples/sshd_config	/etc/ssh/sshd_config	**sshd** (daemon configuration settings)	644	UID(0)
/samples/ssh_config	/etc/ssh/ssh_config	**ssh** client configuration settings	644	UID(0)
/samples/moduli	/etc/ssh/moduli	Diffie-Hellman groups	644	UID(0)
/samples/ssh_prng_cmds	/etc/ssh/ssh_prng_cmds	Commands for gathering entropy	644	UID(0)

Program-generated files

File	Produced by	Description	Permissions	Owner
/var/run/sshd.pid	**sshd**	**sshd** daemon process ID	644	UID(0)

Administrator-generated user files

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, prevent non-superuser **sshd** login and outputs contents to user	644	UID(0)

User-generated files

File	Produced by	Description	Permissions	Owner
$HOME/.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	644	User
$HOME/.ssh/authorized_keys	Copied from $HOME/.ssh/*.pub files of this user's accounts on other (remote) systems	Public keys that can be used to log in to user's account	644	User
$HOME/.rhosts	User	Not recommended. Hosts and users lists to which user can login without password	644	User
$HOME/.shosts	User	Not recommended. Hosts and users lists that users can login via **sshd** only) without password	644	User
$HOME/.ssh/environment	User	User's environment variable initialization at **ssh** login	600	User
$HOME/.ssh/rc	User	User's initialization script at **ssh** login	600	User
$HOME/.ssh/config	User	Copied from **/samples/ssh_config** by user	644	User
$HOME/.ssh/identity	ssh-keygen	User private key file (Protocol 1)	600	User
$HOME/.ssh/id_dsa	ssh-keygen	User private DSA key file	600	User
$HOME/.ssh/id_rsa	ssh-keygen	User private RSA key file	600	User
$HOME/.ssh/identity.pub	ssh-keygen	User public key (Protocol 1)	644	User
$HOME/.ssh/id_dsa.pub	ssh-keygen	User public DSA key	644	User
$HOME/.ssh/id_rsa.pub	ssh-keygen	User public RSA key	644	User

Appendix B. Troubleshooting

This appendix has two sections:
- “Frequently asked questions”
- “Setting up syslogd to debug sshd” on page 165

Frequently asked questions

1. **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 7.
2. **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 3 for information about copying the **sshd_config** file.
3. **sshd will only start in debug mode. If I attempt to start it as a normal production daemon, I see the following error in the syslog: “FOTS1464 Cannot bind any address”.**

   Verify you have all the required service installed. For a discussion of the required service, see *IBM Ported Tools for z/OS Program Directory*.
4. **When I start a new TCP/IP stack in a CINET environment, sshd does not recognize it**.

   If you are running in a CINET environment and you bring up a TCP/IP stack after **sshd** was started, send a SIGHUP signal to the daemon process so it recognizes the new stack. “Steps for starting sshd as a stand-alone daemon” on page 8 mentions the SIGHUP signal.
5. **When I log into z/OS using a non-OpenSSH client, why do some of my keystrokes cause strange behavior, like erasing characters or forcing a new command prompt when I haven’t pressed the Return key?**

   The IETF Secure Shell protocol has the capability for terminal modes to be passed from the client to the server in a portable manner. However, not all SSH clients send terminal modes to the server. In this case, regular EBCDIC keystrokes on z/OS may be misinterpreted by a terminal emulator (which is looking for ASCII code points) as special terminal instructions. To work around this, you can use the **stty** command in a shell profile to define the terminal characteristics for your session.

   Example:

   ```
   stty erase ^H    # ^H is Control-H
   stty quit  ^V    # ^V is Control-V
   stty kill  ^U    # ^U is Control-U
   stty eof   ^D    # ^D is Control-D
   ```
6. **When I run an OpenSSH utility and receive an error, I do not see a message number (for example, FOTSnnnn) associated with it.**

Verify that the NLSPATH environment variable contains ″**/usr/lib/nls/msg/%L/%N.cat**″**.** For more information, see Step 7 on page 5 about setting up the NLSPATH environment variable. If you are running **sshd** and are not seeing message numbers, it could be that the output in question is considered “log” output and may or may not be an error.

7. **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
   ```

8. **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.

9. **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 (eg. /etc/resolv.conf) are working.

10. **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, "~~".

11. **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
    ```

12. **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**. 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.

13. **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.

14. **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.
15. **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.
16. **When I attempt to start ssh, I get the error message "FOTS0944 buffer_get_bignum: input buffer too small"**.

    Your public key or private key file may be corrupted. Regenerate your keys and try again.
17. **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.**

    This error is often seen when the user has **/etc/ssh/sshrc** or **$HOME/.ssh/rc** on the remote host that is generating output to stdout. Make sure that both **/etc/ssh/sshrc** and **$HOME/.ssh/rc** do not produce output to stdout. Write to standard error instead.
18. **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 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 may not have permission to read from it.

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 **syslogd.conf**.

    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** configuration file, add keywords “SyslogFacility” and “LogLevel”. 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 file 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**.

Appendix C. Accessibility

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 Volume 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/Library Server versions of z/OS books in the Internet library at:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

One exception is command syntax that is published in railroad track format; screen-readable copies of z/OS books with that syntax information are separately available in HTML zipped file form upon request to mhvrcfs@us.ibm.com.

Notices

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs that use Open Source Tools for z/OS.

Trademarks

Index

Readers' Comments — We'd Like to Hear from You

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

Publication No. SA22-7985-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SA22-7985-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

SA22-7985-00

Cut or Fold
Along Line

Program Number: 5655-M23

Printed in USA

SA22-7985-00