SSH for OpenVMS Administration and User's Guide

June 2002

This manual provides the system manager with the procedures for installing, managing, and using the SSH for OpenVMS family of software products.

Revision/Update: This is a new manual.

Operating System/Version: OpenVMS VAX V6.2, 7.0, 7.1, 7.2, 7.3;
OpenVMS Alpha V6.2, 7.0, 7.1, 7.2-1, 7.2-2, 7.3

UCX Version: V4.2 and later

TCP/IP Services Version: V5.0 and later

Software Version: 1.0

Process Software
Framingham, Massachusetts
USA

Contents

Chapter 3 Configuring SSH for OpenVMS

Chapter 4 Configuring the Secure Shell (SSH) V1 Server

Chapter 5 Configuring the Secure Shell (SSH) V2 Server

Chapter 6 Accessing Remote Systems with the Secure Shell (SSH) Utilities

Chapter 7 Secure File Transfer

Chapter 8 Monitoring and Controlling SSH

Preface

Introducing This Guide

This guide describes the SSH for OpenVMS software. It covers the following topics: software installation, server and client configuration, server startup and shutdown, using the various SSH clients, and server monitoring and control.

What You Need to Know Beforehand

Before using SSH for OpenVMS, you should be familiar with:

- Computer networks in general
- OpenVMS operating system and file system
- HP's OpenVMS TCP/IP software

How This Guide Is Organized

This guide has the following contents:

- Chapter 1, *Before You Begin*, explains what you need to prepare for an installation.
- Chapter 2, *Installing SSH for OpenVMS*, provides a step-by-step procedure for executing the software installation.
- Chapter 3, *Configuring SSH for OpenVMS*, explains how to configure SSH for OpenVMS.
- Chapter 4, *Configuring the Secure Shell (SSH) V1 Server,* describes how to configure and maintain the SSH for OpenVMS SSH V1 server.
- Chapter 5, *Configuring the Secure Shell (SSH) V2 Server,* describes how to configure and maintain the SSH for OpenVMS SSH V2 server.
- Chapter 6, *Accessing Remote Systems with the Secure Shell (SSH) Utilities*, explains how to configure and maintain the SSH for OpenVMS Secure Shell (SSH) client.

- Chapter 7, *Secure File Transfer*, describes using SCP and FTP over SSH for transferring files in a secure manner.
- Chapter 8, *Monitoring and Controlling SSH*, describes the utilities used for monitoring and controlling the SSH server environment.

Online Help

You can use help at the DCL prompt to find the following:

- Topical help — Access SSH help topics only as follows:

```
$ HELP SSH [topic]
```

The topic entry is optional. You can also enter topics and subtopics at the following prompt and its subprompts:

```
SSH Subtopic?
```

Accessing the SSH for OpenVMS Public Mailing List

Process Software maintains two public mailing lists for SSH for OpenVMS customers:

- **Info-SSH@process.com**
- **SSH-Announce@process.com**

The **Info-SSH@process.com** mailing list is a forum for discussion among SSH for OpenVMS system managers and programmers. Questions and problems regarding SSH for OpenVMS can be posted for a response by any of the subscribers. To subscribe to Info-SSH, send a mail message with the word “SUBSCRIBE” in the body to Info-SSH-request@process.com.

You can retrieve the Info-SSH archives by anonymous FTP to ftp.multinet.process.com. The archives are located in the directory [.MAIL_ARCHIVES.INFO-SSH].

You can also find the Info-SSH archives on the SSH for OpenVMS CD in the [INFO-SSH] directory.

The **SSH-Announce@process.com** mailing list is a one-way communication (from Process Software to you) used for the posting of announcements relating to SSH for OpenVMS (patch releases, product releases, etc.). To subscribe to SSH-Announce, send a mail message with the word “SUBSCRIBE” in the body to SSH-Announce-request@process.com.

Obtaining Customer Support

You can use the following customer support services for information and help about SSH for OpenVMS and other Process Software products if you subscribe to our Product Support Services. (If you bought SSH for OpenVMS products through an authorized Process Software reseller, contact your reseller for technical support.) Contact Technical Support directly using the following methods:

- **Electronic Mail**

  E-mail relays your question to us quickly and allows us to respond, as soon as we have information for you. Send e-mail to **`support@process.com`**. Be sure to include your:

  – Name
  – Telephone number
  – Company name
  – Process Software product name and version number
  – Operating system name and version number

  Describe the problem in as much detail as possible. You should receive an immediate automated response telling you that your call was logged.

- **Telephone**

  If calling within the continental United States or Canada, call Process Software Technical Support toll-free at 1-800-394-8700. If calling from outside the continental United States or Canada, dial +1-508-628-5074. Please be ready to provide your name, company name, and telephone number.

- **World Wide Web**

  There is a variety of useful technical information available on our World Wide Web home page, **`http://www.process.com`** (select **`Customer Support`**).

License Information

SSH for OpenVMS includes a software license that entitles you to install and use it on one machine. Please read and understand the *Software License Agreement* before installing the product. If you want to use SSH for OpenVMS on more than one machine, you need to purchase additional licenses. Contact Process Software or your distributor for details.

Maintenance Services

Process Software offers a variety of software maintenance and support services. Contact us or your distributor for details about these services.

Reader's Comments Page

The *SSH for OpenVMS Administration and User's Guide* includes Reader's Comments as the last page. If you find an error in this guide or have any other comments about it, please let us know. Return a completed copy of the Reader's Comments page, or send e-mail to **techpubs@process.com**.

Please make your comments specific, including page references whenever possible. We would appreciate your comments about our documentation.

Documentation Set

The documentation set for SSH for OpenVMS consists of the following:

- ***Administration and User's Guide*** — For system managers, general users, and those installing the software. The guide provides installation and configuration instructions for the SSH for OpenVMS products.
- ***Online help*** — Topical help, using **HELP SSH *[topic]***
- ***Release Notes*** for the current version of SSH for OpenVMS — For all users, system managers, and application programmers. The *Release Notes* are available online on your SSH for OpenVMS media and are accessible before or after software installation.

Conventions Used

Convention	Meaning
host	Any computer system on the network. The local host is your computer. A remote host is any other computer.
monospaced type	System output or user input. User input is in **`bold type`**. Example: `Is this configuration correct?` **`YES`** Monospaced type also indicates user input where the case of the entry should be preserved.
italic type	Variable value in commands and examples. For example, *username* indicates that you must substitute your actual username. Italic text also identifies documentation references.
[*directory*]	Directory name in an OpenVMS file specification. Include the brackets in the specification.
[optional-text]	(Italicized text and square brackets) Enclosed information is optional. Do not include the brackets when entering the information. Example: **`START/IP line`** ***`address [info]`*** This command indicates that the *info* parameter is optional.
{*value* | *value*}	Denotes that you should use only one of the given values. Do not include the braces or vertical bars when entering the value.
Note!	Information that follows is particularly noteworthy.
CAUTION!	Information that follows is critical in preventing a system interruption or security breach.
`key`	Press the specified key on your keyboard.
`Ctrl/key`	Press the control key and the other specified key simultaneously.
`Return`	Press the Return or Enter key on your keyboard.

Chapter 1

Before You Begin

Introduction

This chapter introduces you to and prepares you for SSH product installation, configuration, startup, and testing. It is for the OpenVMS system manager or technician responsible for product installation and configuration.

Steps to Get SSH Up and Running

To get SSH up and working, you must perform the following steps:

Table 1-1 Getting SSH Up and Running

1	Load the license pack.	
2	Install the software.	See Chapter 2, *Installing SSH for OpenVMS*
3	Configure the SSH for OpenVMS environment.	See Chapter 3, *Configuring SSH for OpenVMS*
4	Configure the SSH for OpenVMS SSH V1 server.	See Chapter 4 *Configuring the Secure Shell (SSH) V1 Server*
5	Configure the SSH for OpenVMS SSH V2 server.	See Chapter 5, *Configuring the Secure Shell (SSH) V2 Server*
6	Configure the SSH for OpenVMS client.	See Chapter 6, *Accessing Remote Systems with the Secure Shell (SSH) Utilities*

Prepare for Installation

SSH for OpenVMS installation involves using the VMSINSTAL procedure. Preparing for installation involves:

- Understanding the hardware and software requirements
- Determining if you have sufficient disk space and global pages for the installation
- Determining where to install the software

Hardware Requirements

SSH for OpenVMS has no special hardware requirements beyond those stated in the Software Product Description for HP's TCP/IP Services.

Software Requirements

SSH for OpenVMS supports OpenVMS/VAX version 6.2, 7.0, 7.1, 7.2, 7.3; OpenVMS Alpha version 6.2, 7.0, 7.1, 7.2-1, 7.2-2, 7.3; UCX version 4.2 and later, and TCP/IP Services version 5.0 and later.

Disk Space and Global Pages

The destination device for your SSH for OpenVMS software must have enough disk space so that you can install and run the software. If you plan to install every SSH for OpenVMS component, your system must have approximately the following values.

System	Number of Blocks Needed to Install SSH for OpenVMS	Number of Blocks Needed After Installation
VAX	200,000	about 34,000
Alpha	200,000	about 46,000

The runtime values are slightly higher once you configure and start SSH for OpenVMS.

You should also have at least 50,000 free global pages (GBLPAGES) on your system before installing SSH for OpenVMS. Use SHOW GBLPAGES in the SYSGEN utility to determine the parameter value and change it using SET GBLPAGES if necessary.

Insufficient GBLPAGES can abort the installation and leave your system command tables disconnected. The only way to recover is through a system reboot.

General Requirements

Check at this point that you:

- Have OPER, SYSPRV, or BYPASS privileges
- Can log in to the system manager's account
- Are the only user logged in (recommended)
- Backed up your system disk on a known, good, current, full backup (recommended)
- Need to reinstall SSH for OpenVMS after performing a major VMS upgrade
- If SSH for OpenVMS is currently running, shut it down. This is mandatory.
- Ensure TCP/IP Services (or UCX) is currently running.

Where to Install SSH for OpenVMS

Install SSH for OpenVMS in a location depending on the following:

- Generally, on your system disk, but you can install SSH for OpenVMS anywhere, just answer the question when it appears. This is also where you would keep your "common" files. Node-specific files should always be on your system disk.
- If the machine is in a single platform cluster, on a common disk.
- If the machine is in a mixed platform cluster, once on the Alpha system disk (or disks) and once on the VAX common system disk.

Release Notes and Online Documentation

The SSH for OpenVMS *Release Notes* provide important information on the current release. If you are installing from CD-ROM, you can access the *Release Notes* and the full SSH for OpenVMS documentation as PDF files. They are in the [DOCUMENTATION] directory. The *Release Notes* for this product are in the file SSH010.RELEASE_NOTES. The other file (MULTINET044.RELEASE_NOTES) may be ignored.

- If you are installing from disk, you can read or print the *Release Notes* as a text file, which you can obtain in one of three ways:
  - By performing a partial installation
  - During the full installation
  - After the installation

To perform a partial installation (see Example 1-1):

1 Invoke VMSINSTAL at the system prompt:

$ **@SYS$UPDATE:VMSINSTAL MULTINET044** ***device*** **OPTIONS N**

The *device* is the mount location of the distribution volumes.

2 Press **Return** at the prompt

```
Are you satisfied with the backup of your system disk [YES]?.
```

3 Select the option by number as to whether you want to display or print the *Release Notes*, or both.

4 If you requested a printout, enter the queue name for the printer. The default is SYS$PRINT.

5 Press **Return** at the prompt

```
Do you want to continue the installation [NO]?:.
```

(Note that if you enter YES at the prompt, you proceed with the full installation.)

6 You see the message

```
Product's release notes have been moved to SYS$HELP.
```

7 If you want to read or print the *Release Notes* after you exit the installation, you can access the MULTINET044.RELEASE_NOTES and SSH010.RELEASE_NOTES files in the SYS$HELP directory, as in:

```
$ TYPE SYS$HELP:MULTINET044.RELEASE_NOTES
```

or

```
$ TYPE SYS$HELP:SSH010.RELEASE_NOTES
```

Note! For this command to work as desired, do not redefine the SYS$HELP directory logical.

Example 1-1 Performing a Partial Installation to Obtain the *Release Notes*

```
$ @SYS$UPDATE:VMSINSTAL MULTINET044 DKA300: OPTIONS N        [1]

        OpenVMS AXP Software Product Installation Procedure V7.1
It is 3-JUNE-2002 at 11:01.
Enter a question mark (?) at any time for help.
* Are you satisfied with the backup of your system disk [YES]? Return   [2]
The following products will be processed:
  MULTINET V4.4
         Beginning installation of MULTINET V4.4 at 11:01
%VMSINSTAL-I-RESTORE, Restoring product save set A ...
    Release notes included with this kit are always copied to SYS$HELP.
Additional Release Notes Options:
        1.  Display release notes
        2.  Print release notes
        3.  Both 1 and 2
        4.  None of the above

* Select option [2]: Return      [3]

* Queue name [SYS$PRINT]: Return      [4]
Job MULTINET044 (queue SYS$PRINT, entry 1) started on SYS$PRINT

* Do you want to continue the installation [NO]? Return     [5]
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS$HELP.
        VMSINSTAL procedure done at 11:02

    .
```

```
.
.
$ TYPE SYS$HELP:MULTINET044.RELEASE_NOTES        [6]

$TYPE SYS$HELP:SSH010.RELEASE_NOTES              [7]
```

Chapter 2

Installing SSH for OpenVMS

Introduction

This chapter takes you through the SSH for OpenVMS product installation procedure and certain post-installation tasks. It is for the OpenVMS system manager, administrator, or technician responsible for product installation.

To prepare for installation, see Chapter 1, *Before You Begin*.

Note! Once you have installed SSH for OpenVMS, you need to reinstall it after you have done a major VMS upgrade.

To install SSH for OpenVMS:

1 Load the software.

2 Run the VMSINSTAL procedure.

3 Install other products, if needed, and perform post-installation tasks.

Load the Software

SSH for OpenVMS is shipped to you on CD-ROM media.

There are three steps to loading the SSH for OpenVMS software:

1 Log in to the system manager's account.

2 If SSH for OpenVMS is currently running, shut it down:

```
$ SSHCTRL SHUTDOWN
```

If you are installing on a VMScluster, shut down SSH for OpenVMS on each node in the cluster.

3 Physically load the distribution media onto the appropriate device.

- In a VMScluster environment, if you want to access the media from more than one node, enter the following:

$ **MOUNT/CLUSTER/SYSTEM *device MULTINET044***

- On a standalone system, or if you want to prevent multiple users from accessing the software, enter the following:

$ **MOUNT *device MULTINET044***

Note! If you install SSH for OpenVMS on a VMS cluster that has a common system disk, install the software on only one node in the cluster. **If reinstalling or upgrading SSH for OpenVMS, first shut down SSH for OpenVMS on all nodes in the cluster**.

Be sure to configure SSH for OpenVMS on all systems in a VMS cluster that has a common system disk, even though it only needs to be installed once.

Start VMSINSTAL

VMSINSTAL is the VMS installation program for layered products. VMSINSTAL prompts you for any information it needs. Table 2-1 shows the steps to follow.

Table 2-1 Starting VMSINSTAL

Step	For this task...	Enter this response...
1	Make sure that you are logged in to the system manager's account, and invoke VMSINSTAL	@SYS$UPDATE:VMSINSTAL
2	Determine if you are satisfied with your system disk backup	**Return** or **Y** (Yes) or **N** (No)
3	Determine where the distribution volumes will be mounted	The disk (and directory) or tape device where you want the software to be mounted.
4	Enter the products you want processed from the first distribution volume set	**MULTINET044**
5	Enter the installation options you wish to use (such as obtaining the *Release Notes*)	**Return** for no options or **N** for *Release Notes*.
6	Specify the directory where you want the files installed.	**Return** if accepting default of SYS$SYSDEVICE:[MULTINET]
7	Specify the directory where you want the system-specific files installed	**Return** if accepting default of [<nodename>];e.g, [ROSES]
8	Specify that you are performing an SSH for OpenVMS installation.	Enter "Y" to install SSH for VMS.

Sample Installation

```
$ @SYS$UPDATE:VMSINSTAL MULTINET044 DEVICE:[MULTINET044]

        OpenVMS AXP Software Product Installation Procedure V7.3


It is 10-JUN-2002 at 12:21.

Enter a question mark (?) at any time for help.


%VMSINSTAL-W-ACTIVE, The following processes are still active:
TCPIP$FTP_1

* Do you want to continue anyway [NO]? y
* Are you satisfied with the backup of your system disk [YES]?


The following products will be processed:

  MULTINET V4.4


        Beginning installation of MULTINET V4.4 at 11:29

%VMSINSTAL-I-RESTORE, Restoring product save set A ...
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to
SYS$HELP.

* Where do you want to install MultiNet [SYS$SYSDEVICE:[MULTINET]]:
* What do you want to call the system-specific directory [PHNTOM]:

    You may perform one of the following:

      - Full MultiNet installation
      - SSH for OpenVMS installation

    Note that you must have the appropriate license for
   the selection you make.

* Are you doing an SSH for OpenVMS installation only (YES/NO) [N]? Y

                    SSH for OpenVMS
                    MultiNet (R)

ALL RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES

This licensed material is the valuable property of Process Software.
Its use, duplication, or disclosure is subject to the restrictions set
forth in the License Agreement.

Other use, duplication or disclosure, unless expressly provided for in

the license agreement, is unlawful.
```

```
Installing SSH for OpenVMS V1.0 Rev A
* Do you want to install the online documentation [YES]?

   The HTML documentation requires 700 blocks.

* Do you want to install the HTML documentation [YES]?

   The PDF documentation requires 1640 blocks.

* Do you want to install the PDF documentation [YES]?

    The SSH for OpenVMS software will be installed with these
    selected components:

    * Online documentation

        - HTML Documentation

        - PDF Documentation

* Would you like to change your selections [NO]?
* Do you want to purge files replaced by this installation [YES]?
* Configure SSH for OpenVMS after installation [NO]?
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.HELP].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$COMMON_ROOT:[MULTINET].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$COMMON_ROOT:[MULTINET.HELP].

The installation will now proceed with no further questions.

%VMSINSTAL-I-RESTORE, Restoring product save set Y ...
%MULTINET-I-INSTALLING, Installing SSH for OpenVMS files
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$COMMON_ROOT:[MULTINET.PSCSSH].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.PSCSSH].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.PSCSSH.LOG].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2.HOSTKEYS].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2.KNOWNHOSTS].
%MULTINET-I-CREATING, Creating SSH for OpenVMS startup file
```

```
      ****************************************************************
      *                                                              *
      *                                                              *
      *                                                              *
      *  To start SSH for OpenVMS, add the following line to your    *
      *                                                              *
      *  SYSTARTUP_VMS.COM file after you have configured SSH for    *
      *                                                              *
      *  OpenVMS:                                                    *
      *                                                              *
      *                                                              *
      *             $ @SYS$STARTUP:PSCSSH$STARTUP                    *
      *                                                              *
      *                                                              *
      ****************************************************************

%MULTINET-I-INSTALLING, Installing the online documentation files
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$SPECIFIC_ROOT:[MULTINET.PSCSSH.DOCUMENTS].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
MU$COMMON_ROOT:[MULTINET.PSCSSH.DOCUMENTS].
%MULTINET-I-INSTALLING, Installing SSH for OpenVMS HELP library
%MULTINET-I-DELETING, Deleting obsolete MultiNet files
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target
directories...

        Installation of MULTINET V4.4 completed at 12:33

    Adding history entry in VMI$ROOT:[SYSUPD]VMSINSTAL.HISTORY

    Creating installation data file:
VMI$ROOT:[SYSUPD]MULTINET044.VMI_DATA
```

Installing SSH for OpenVMS on a Common VMScluster System Disk

After installing SSH for OpenVMS on one node of a VMScluster with a common system disk, you must perform the following steps on each additional cluster node that shares the common system disk:

1. Log in (telnet/set host/etc.) to the next node of the cluster.
2. Create the MULTINET logicals by using the following command:
   $ **@SYS$STARTUP:PSCSSH$STARTUP LOGICALS**
3. Make the node-specific SSH root and configure SSH for this node:
   $ **@MULTINET:SSH_MAKE_ROOT**

4. Start SSH for OpenVMS:
   ```
   $ @SYS$STARTUP:PSCSSH$STARTUP
   ```
5. Repeat steps 1-4 for each remaining node of the cluster except for the one where SSH was originally installed.

Installing SSH for OpenVMS on Mixed Platform Clusters

SSH for OpenVMS has no files which can be shared between cluster systems of different architectures.

Chapter 3

Configuring SSH for OpenVMS

Introduction

This chapter describes how to configure the SSHD Master process, which controls access to the SSH servers for the SSH for OpenVMS software.

For a basic configuration, accept the default values for each component, which appear after a prompt. This also helps you step through the process more quickly.

After performing the basic configuration, you must perform the advanced configuration for the SSH1 and SSH2 servers, and for the SSH clients as desired. Chapters 4 through 7 describe the configuration and use of these components.

The SSH Configuration Utility

SSH is the Secure Shell protocol. SSH for OpenVMS provides support for both SSH Version 1 protocol and SSH Version 2 protocol.

Please note that in addition to the configuration performed via CNFSSH as described below, there are configuration files for both the SSH1/SSH2 servers and SSH client which must be modified as appropriate to meet the security requirements of your organization. Refer to chapters 4 and 5 of this manual for details on the configuration files.

You can enable the SSH utility as shown in Example 3-1.

Example 3-1 Using the SSH Utility

```
$ @MULTINET:CNFSSH CONFIGURE

SSH for OpenVMS Version V1.0A SSH Configuration procedure

This procedure helps you define the parameters needed to get SSH for
OpenVMS running on this system.

This procedure creates the configuration data file,
MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH]SSH_CONFIGURE.COM,to reflect
your system's configuration.

For detailed information on the following parameters, refer to the SSH
for OpenVMS Administration and User's Guide.

SSH for OpenVMS supports both SSH1 and SSH2 servers.  You may configure
SSH for OpenVMS to support either SSH1 servers or SSH2 servers, or both.
Note that the choice of either or both servers has no impact on the SSH
for OpenVMS client, which supports both SSH1 and SSH2 remote servers.

Do you want to enable the SSH1 server [NO]?YES
Do you want to enable the SSH2 server [NO]?YES

For SSH1, you must specify the number of bits in the RSA key.  The range
is 512 to 32768 bits, but keys longer than 1024 are generally not much
safer, and they significantly increase the amount of CPU time consumed
by key generation when the SSHD_MASTER process is starting.

Enter the number of bits in the RSA key [768]:

You may specify an alternate configuration file for the SSH1 server. If
you have already specified an alternate config file, enter a single
space and hit RETURN at the prompt to reset it to the default file name.

Enter an alternate SSH1 configuration filename []:

You may specify an alternate configuration file for the SSH2 server. If
you have already specified an alternate config file, enter a single
space and hit RETURN at the prompt to reset it to the default file name.

Enter an alternate SSH2 configuration filename []:


Specify the level of debug for the SSH1 and SSH2 servers.

For SSH1, any non-zero value will turn on debug, but there is no "degree
of debug".

For SSH2, this is a value from 0 to 50, where zero is no debug and 50 is
the maximum level of debug.  Note that at levels exceeding debug level
8,there may be a substantial impact on SSH2 server (and possibly, the
system,too) performance due to the amount of information logged.
```

```
Enter the debug level [0 - 50]:

For SSH1, you may enter the name of an alternate RSA host key file.  If
you have already specified an alternate host key file, enter a single
space and hit RETURN at the prompt to reset it to the default file name.

Enter an alternate SSH1 public server host key file []:

Specify the time in seconds after which the server private key is
generated. This is only done for SSH1 sessions.

Enter the key regeneration time [3600]:

You may specify the number of seconds a user has to enter a password
during user authentication (default = 0).  In addition, you may allow
this to default to the value used by OpenVMS when a user is logging into
a non-SSH session.  To specify an infinite wait time, enter 0 for the
timeout value.

Do you want to change the default login grace time [NO]?

Specify the port for the SSH server to listen on, if you wish to use a
port other than the default port of 22.

Enter port to use [22]:
Do you want any messages logged by the SSH server at all [YES]?
Do you want verbose logging by the SSH server [NO]?

You may specify the maximum number of concurrent SSH sessions to be
allowed on the server.  This is the total of both SSH1 and SSH2
sessions. The default is 1000 sessions.

Enter maximum number of concurrent SSH sessions [1-1000, 1000]:

In OpenVMS, users with passwords that have expired because the SYSUAF
PWDLIFETIME value has been exceeded are allowed to log into the system,
and are then forced to change their password.  The SSH1 protocol does
not allow for that condition.  Answer "YES" to the following question if
you wish to allow users with expired passwords to still log into the
system.  They WILL NOT be forced to change their password.

Note that the SSH2 protocol is not restricted as the SSH1 protocol is;
changing of expired passwords, save for pre-generated passwords, is
performed by many SSH2 clients (including the SSH for OpenVMS client).

Do you want to allow users with expired passwords to log in [NO]?

In OpenVMS, users with passwords that have been pre-expired by the
system manager are allowed to log into the system, and are then forced
to change their password.  The SSH1 protocol does not allow for that
condition.  Answer "YES" to the following question if you wish to allow
users with pre-expired passwords to still log into the system.  They
WILL NOT be forced to change their password.

Note that the SSH2 protocol is not restricted as the SSH1 protocol is;
```

```
changing of expired passwords, save for pre-generated passwords, is
performed by many SSH2 clients (including the SSH for OpenVMS client).

Do you want to allow users with preexpired passwords to log in [NO]?

The SSH1 protocol does not permit the display of the contents of the
SYS$ANNOUNCE logical or file prior to a user logging in.  Answering "Y"
to the next question will cause the SSH for OpenVMS client to display
the contents of SYS$ANNOUNCE after user authentication is completed but
before the contents of SYS$WELCOME are displayed.

Do you want to display SYS$ANNOUNCE [NO]?

When generating user keys, a passphrase may be used to further protect
the key.  No limit is normally enforced for the length of the
passphrase. However, you may specify a minimum length the passphrase may
be.

What you want the minimum passphrase length to be for SSH1 [0-1024, 0]?
What you want the minimum passphrase length to be for SSH2 [0-1024, 0]?

The SSH1 host key has not yet been generated.  Answer YES to the
following question to generate the key now.  Answer NO to generate
the key manually later by issuing the command:

    $ MULTINET SSHKEYGEN /SSH1/HOST

Generating a host key can take a few minutes on slow systems.

Do you want to generate the SSH1 host key now [Y]?
Initializing random number generator...
Generating p:  .................++ (distance 238)
Generating q:  ..................................................++
(distance 842)
Computing the keys...
Testing the keys...
Key generation complete.
Key file will be MULTINET_ROOT:[MULTINET.PSCSSH.SSH]SSH_HOST_KEY.
Your identification has been saved in
MULTINET_ROOT:[MULTINET.PSCSSH.SSH]SSH_HOST_KEY..
Your public key is:
1024 33
158219526854708373223273549671898538484019382056540756180743251896008268
4836722491925723206793361916371976479312524684849247423817691927521755299
999742062
804079403652395183296863957945714446720630016910346731983816732024731065
63769428
303384286498131699887049314519433804844962219668666235774358798424562221
57799
SYSTEM@rose.flower.com
Your public key has been saved in
MULTINET_ROOT:[MULTINET.PSCSSH.SSH]SSH_HOST_KEY.pub
```

```
The SSH2 host key has not yet been generated.  Answer YES to the
following question to generate the key now.  Answer NO to generate the
key manually later by issuing the command:

    $ MULTINET SSHKEYGEN /SSH2/HOST

Generating a host key can take a few minutes on slow systems.

Do you want to generate the SSH2 host key now [Y]?
Generating 1024-bit dsa key pair
    3 o.oOo.oOo.oo
Key generated.
1024-bit dsa, dilbert@rose.flower.com, Thu May 02 2002 08:21:41
Private key saved to multinet_ssh2_hostkey_dir:hostkey.
Public key saved to multinet_ssh2_hostkey_dir:hostkey.pub

SSH Configuration completed.

Review the additional steps you may need to perform as described in the
configuration chapters of the SSH for OpenVMS Administration and User's
Guide before starting SSH.

Refer to the "Monitoring and Controlling SSH" chapter of the SSH for
OpenVMS Administration and User's Guide for information on starting SSH.
```

Chapter 4

Configuring the Secure Shell (SSH) V1 Server

This chapter describes how to configure and maintain the SSH for OpenVMS SSH1 server.

This is the server side of the software that allows secure interactive connections from other computers in the manner of rlogin/rshell/telnet. The SSH server has been developed to discriminate between SSH v1 and SSH v2 protocols, so the two protocols can coexist simultaneously on the same system.

SSH1 and SSH2 Differences

SSH1 and SSH2 are different, and incompatible, protocols. The SSH1 implementation is based on the V1.5 protocol and 1.3.7 F-Secure code base, and the SSH2 implementation is based on the V2 protocol and the F-Secure 3.1.0 code base. While SSH2 is generally regarded to be more secure than SSH1, both protocols are offered by SSH for OpenVMS, and although they are incompatible, they may exist simultaneously on a system. The server front-end identifies what protocol a client desires to use, and will create an appropriate server for that client.

Restrictions:

When using SSH to connect to a VMS server, if the VMS account is set up with a secondary password, SSH does not prompt the user for the secondary password. If the VMS primary password entered is valid, the user is logged in, bypassing the secondary password.

When using SSH to execute single commands (in the same manner as RSHELL), some keystrokes like **CTRL/Y** are ignored. In addition, some interactive programs such as HELP may not function as expected. This is a restriction of SSH. If this behavior poses a problem, log into the remote system using SSH in interactive mode to execute the program.

Understanding the Secure Shell Server

The Secure Shell daemon (SSHD) is the daemon program for SSH that listens for connections from clients. The server program replaces rshell and telnet programs. The server/client programs provide secure encrypted communications between two untrusted hosts over an insecure network. A new daemon is created for each incoming connection. These daemons handle key exchange, encryption, authentication, command execution, and data exchange.

Servers and Clients

An SSH server is an OpenVMS system that acts as a host for executing interactive commands or for conducting an interactive session. The server software consists of two pieces of software (for future reference, “SSHD” will refer to both SSHD_MASTER and SSHD, unless otherwise specified):

- SSHD_MASTER, recognizes the differences between SSH v1 and SSH v2 and starts the appropriate server. If the request is for SSH v1, then the SSH v1 server is run; if the request is for SSH v2, then the SSH v2 server is run.
- SSHD, a copy of which is spawned for each connection instance. SSHD handles all the interaction with the SSH client.

A client is any system that accesses the server. A client program (SSH) is provided with SSH for OpenVMS, but any SSH client that uses SSH version 1 protocol may be used to access the server. Examples of such programs are SSH for OpenVMS, MultiNet SSH, TCPware SSH, and FISSH on OpenVMS; SecureCRT and TTSSH on Windows®-based systems; and other SSH programs on UNIX-based systems.

Security

Each host has a host-specific RSA key (normally 1024 bits) that identifies the host. Additionally, when the SSHD daemon starts, it generates a server RSA key (normally 768 bits). This key is regenerated every hour (the time may be changed in the configuration file) if it has been used, and is never stored on disk. Whenever a client connects to the SSHD daemon,

- SSHD sends its host and server public keys to the client.
- The client compares the host key against its own database to verify that it has not changed.
- The client 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.
- The client and the server start to 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, IDEA (the default), DES, 3DES, BLOWFISH, and ARCFOUR are supported.

- The client selects the encryption algorithm to use from those offered by the server.
- The server and the 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
- password-based authentication

Note! 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 the RLOGIN and RSHELL services are disabled.

If the client authenticates itself successfully, a dialog is entered for preparing the session. At this time the client may request things like:

- forwarding X11 connections
- forwarding TCP/IP connections
- forwarding the authentication agent connection over the secure channel

Finally, the client either requests an interactive session or execution of a command. The client and the server enter session mode. In this mode, either the client or the server may send data at any time, and such data is forwarded to/from the virtual terminal or command on the server side, and the user terminal in 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.

Options

The SSHD Master process is configured via the CNFSSH command procedure. This procedure creates the SSH_DIR:SSH_CONFIGURE.COM file, and this is used when starting up the SSHD Master process. Once CNFSSH is used to modify the SSHD Master options, SSH should be restarted using @SYS$STARTUP:PSCSSH$STARTUP.COM.

Note! The recommended method to start the SSHD Master process is to use the @SYS$STARTUP:PSCSSH$STARTUP command. All of these options are set using @MULTINET:CNFSSH CONFIGURE

Configuration File

The individual SSHD server processes read configuration data from SSH_DIR:SSHD_CONFIG (or the file specified via the CNFSSH procedure). The file contains keyword value pairs, one per line. Lines starting with ‘#’ and empty lines are interpreted as comments. The following keywords are possible. Keywords are case insensitive.

Note! HP’s TCP/IP services do not use the traditional UNIX rhosts and hosts.equiv files; it uses a proprietary format. Therefore, any information added to HP’s files via the “ADD PROXY” command must also be manually added to the SSH_DIR:RHOSTS and SSH_DIR:HOSTS.EQUIV files in order for it to be used by SSH for OpenVMS.

Table 4-1 Configuration File Keywords [SSHD_CONFIG]

Keyword	Value	Default	Description
AllowForwardingPort	Port List		Can be followed by any number of port numbers, separated by spaces. Remote forwarding is allowed for those ports whose number matches one of the patterns. You can use '*' as a wildcard entry for all ports. You can use these formats '>x', '<x', and 'x_x' to specify greater than, less than, or inclusive port range. By default, all port forwardings are allowed.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
AllowForwardingTo	Host/port list		Can be followed by any number of hostname and port number patterns, separated by spaces. A port number pattern is separated from a hostname pattern by a colon. Forwardings from the client are allowed to those hosts and port pairs whose name and port number match one of the patterns. You can use '*' and '?' as wildcards in the patterns for host names. Normal name servers are used to map the client's host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as the hostname. You can use '*' as a wildcard entry for all ports. You can use these formats '>x', '<x', and 'x_x' to specify greater than, less than, or inclusive port range. By default, all port forwardings are allowed.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
AllowGroups	List		Can be followed by any number of OpenVMS rights identifier patterns, separated by spaces. Login is allowed only if the user's list of rights identifiers contains an identifier that matches one of the patterns. You can use '*' and '?' as wildcards in the patterns. By default, logins as all users are allowed. ***Note!*** All other login authentication steps must be completed successfully. DenyGroups is an additional restriction.
AllowHosts	Host list		Can be followed by any number of host name patterns, separated by spaces. Login is allowed only from hosts whose name matches one of the patterns. You can use '*' and '?' as wildcards in the patterns. Normal name servers are used to map the client's host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as the hostname. By default, all hosts are allowed to connect.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
AllowSHosts	Host list		Can be followed by any number of host name patterns, separated by spaces. .SHOSTS (and .RHOSTS and SSH_DIR:HOSTS.EQUIV) entries are honored for hosts whose name matches one of the patterns. Servers are used to map the client's host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as the host name. By default, all hosts are allowed to connect.
AllowTcpForwarding	Y/N	Y	Specifies whether TCP forwarding is permitted. ***Note!*** Disabling TCP forwarding does not improve security in any way, as users can always install their own forwarders.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
AllowUsers	User list		Can be followed by any number of user name patterns or user@host patterns, separated by spaces. Host name may be either the DNS name or the IP address. Login is allowed only for users whose name matches one of the patterns. You can use '*' and '?' as wildcards in the patterns. By default, logins as all users are allowed. ***Note!*** All other login authentication steps must be completed successfully. DenyUsers is an additional restriction.
DenyForwardingPort	Port list		Can be followed by any number of port numbers, separated by spaces. Remote forwardings are disallowed for those ports whose number matches one of the patterns. You can use '*' as a wildcard entry for all ports. You can use these formats '>x', '<x', and 'x_x' to specify greater than, less than, or inclusive port range.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
DenyForwardingTo	Host/port list		Can be followed by any number of hostname and port number patterns, separated by spaces. A port number pattern is separated from a hostname by a colon. Forwardings from the client are disallowed to those hosts and port pairs whose name and port number match one of the patterns. You can use '*' and '?' as wildcards in the patterns for host names. Normal name servers are used to map the client's host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as a host name. You can use '*' as a wildcard entry for all ports. You can use these formats '>x', '<x', and 'x_x' to specify greater than, less than, or inclusive port range.
DenyGroups	Rights list		Can be followed by any number of OpenVMS rights identifier patterns, separated by spaces. Login is disallowed only if the user's list of rights identifiers contains an identifier that matches one of the patterns.
DenyHosts	Host list		Can be followed by any number of host name patterns, separated by spaces. Login is disallowed from the host whose name matches any of the patterns.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
DenySHosts	Host list		Can be followed by any number of host name patterns, separated by spaces. .SHOSTS (and .RHOSTS and SSH_DIR:HOSTS.EQUIV) entries whose name matches any of the patterns are ignored.
DenyUsers	User list		Can be followed by any number of user name patterns or user@host patterns, separated by spaces. A host name may be either the DNS name or the IP address. Login is disallowed for a user whose name matches any of the patterns.
HostKey	Filename	SSH_HOST_KEY	Specifies the file containing the private key.
IdleTimeout	Time	0	Sets the idle timeout limit in: • seconds (s or nothing after the number) • minutes (m) • hours (h) • days (d) • weeks (w) If the connection has been idle (all channels) for the time specified, the process is terminated and the connection is closed. An idle process is one that has done no I/O to stdin or stdout in the timeout value. The default value of 0 (zero) for IdleTimeout means no timeout.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
IgnoreRhosts	Y/N	N	Specifies that the `SYS$LOGIN:RHOSTS` file will not be used in authentication. `SSH_DIR:HOSTS.EQUIV` is still used.
KeepAlive	Y/N	Y	Specifies whether the system should send keepalive messages to the other side. If sent, termination of the connection or crash of one of the machines will be noticed. This means that connections will terminate if the route is down temporarily. If keepalives are not ended, sessions may hang indefinitely on the server, leaving “ghost” users and consuming server resources. To disable keepalives, set the value to no in both the server and the client configuration files.
KeyRegenerationInterval	Time	3600	Specifies how long to wait before the server key is regenerated automatically. Regeneration prevents decrypting captured sessions by later breaking into the machine and stealing the keys. The key is never stored on disk. If the value is 0, the key is never regenerated.
ListenAddressee			Specifies the IP address of the interface where the SSHD server socket is BIND.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
LoginGraceTime	Time	600	Specifies the time the server should disconnect if the user has not logged in successfully. If the value is 0, there is no time limit. The default is 600 seconds.
PasswordAuthentication	Y/N	Y	Specifies whether password authentication is allowed. The default is yes.
PermitEmptyPasswords	Y/N	N	Specifies whether the server allows login to accounts with empty password strings when password authentication is allowed. The default is no. ***Note!*** Use of this keyword may contribute to your system becoming insecure. Process Software encourages you to NOT enable the use of empty passwords.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
PermitRootLogin	Y/N	N	Specifies whether the user can log in as SYSTEM using SSH. This keyword may be set to: yes — allows SYSTEM logins through any of the authentication types allowed for other users. no (default) — disables SYSTEM logins through any of the authentication methods (nopwd and no are equivalent), unless you have a rhosts or `SYS$DISK:[<login_dir>.SSH]AUTHORIZED_KEYS` file in the SYS$MANAGER directory. nopwd — disables password-authenticated SYSTEM logins. System login with RSA authentication when the "command" option has been specified is allowed regardless of the value of this keyword (which may be useful for taking remote backups even if SYSTEM login is not allowed).
QuietMode	Y/N	N	Specifies whether the system runs in quiet mode. In quiet mode, nothing is logged in the system log, except fatal errors.
RandomSeed	Filename	Random_seed	Specifies the SSH:DIR file containing the random seed for the server. This file is created automatically and updated regularly.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
RhostsAuthentication	Y/N	N	Specifies whether authentication using `SYS$LOGIN:RHOSTS` or `SSH_DIR:HOSTS.EQUIV` files is sufficient. Normally, this method should not be permitted because it is insecure. Use RhostsRSAAuthentication because it performs RSA-based host authentication in addition to normal rhosts or `SSH_DIR:HOSTS.EQUIV` authentication. The default is no.
RhostsRSAAuthentication	Y/N	Y	Specifies whether `SYS$LOGIN:RHOSTS` or `SSH_DIR:HOSTS.EQUIV` authentication together with successful RSA host authentication is allowed.
RSAAuthentication	Y/N	Y	Specifies whether pure RSA authentication is allowed.
SilentDeny	Y/N	Y	Specifies whether denied (or not allowed) connections are denied silently (just close the connection, no logging, etc.) or if they are closed cleanly (send an error message and log the connection attempt). Defaults to silent mode, yes.
StrictIntrusionLogging	Y/N	Y	Log intrusion attempts for all failed authentication methods.
StrictModes	Y/N	Y	Specifies whether SSH should check file protection and ownership of the user's home directory and rhosts files before accepting login.

Table 4-1 Configuration File Keywords [SSHD_CONFIG] (Continued)

Keyword	Value	Default	Description
SyslogFacility	Syslog level	"AUTH"	Gives the facility code that is used when logging messages from SSHD. Any valid syslog facility code may be used.
X11DisplayOffset	#offset	10	Specifies the first display number available for SSHD's X11 forwarding. This prevents SSHD from interfering with real X11 servers.

Starting the SSH Server for the First Time

Follow these instructions to configure the SSH server. If SSH isn't currently running, you must define the MULTINET logicals by using:

```
$ @SYS$STARTUP:PSCSSH$STARTUP LOGICALS
```

1 Use the CNFSSH utility to configure the SSH server.

Note! SSH for OpenVMS must be running before issuing the SSHKEYGEN command.

2 Use SSHKEYGEN to create the file SSH_HOST_KEY in the SSH_DIR: directory if it has not been created as a result of executing @MULTINET:CNFSSH CONFIGURE.

```
$ MULTINET SSHKEYGEN/SSH1/HOST
Initializing random number generator...
Generating p:  ...++ (distance 64)
Generating q:  ....................................++ (distance 516)
Computing the keys...
Testing the keys...
Key generation complete.
Key file will be MULTINET_ROOT:[MULTINET.PSCSSH.SSH]SSH_HOST_KEY
Your identification has been saved in
MULTINET_ROOT:[MULTINET.PSCSSH.SSH]SSH_HOST_KEY
Your public key is:
1024 37
121031836557669869786536786929196947638822844496990561186427630 8
907277690446274441596682102010946361764420239729464227794671854 9
440444257759486829708717101335974385318244257992380130202084401 1
534375490984751397316024932473591314633023241042493675101595361 1
187168724911238579405373228915848504593196961275605927
SYSTEM@gg1.prr.com
Your public key has been saved in
MULTINET_ROOT:[MULTINET.PSCSSH.SSH]SSH_HOST_KEY.PUB
```

3 Edit the default configuration file at SSH_DIR:SSHD_CONFIG (if you wish to change the default settings). This default configuration is the same as contained in the file MULTINET:SSHD_CONFIG.TEMPLATE

Note! As delivered, the template file provides a reasonably secure SSH environment. However, Process Software recommends this file be examined and modified appropriately to reflect the security requirements of your organization.

4 Restart SSH. This creates the SSH server process and defines the SSH logical names.

```
$ SSHCTRL RESTART
$ SHOW PROCESS "SSHD Master"

7-JUN-2002 09:03:06.42  User: SYSTEM      Process ID:   00000057
                        Node: PANTHR      Process name: "SSHD Master"

Terminal:
User Identifier:    [SYSTEM]
Base priority:      4
Default file spec:  Not available
Number of Kthreads: 1

Devices allocated:  BG1:
                    BG2:

$ SHOW LOGICAL/SYSTEM SSH*

(LNM$SYSTEM_TABLE)

  "SSH_DIR" = "MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH]"
  "SSH_EXE" = "MULTINET_COMMON_ROOT:[MULTINET.PSCSSH.SSH]"
  "SSH_LOG" = "MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.LOG]"
  "SSH_MAX_SESSIONS" = "100"
  "SSH_TERM_MBX" = "MBA36:"
```

Changing SSH Configuration File After Enabling SSH

If you make a change to the SSH configuration file after you have enabled SSH, you have to restart SSH. To have the changes take effect, use the command:

```
$ SSHCTRL RESTART
```

Connection and Login Process

To create a session, SSHD does the following:

1. SSHD_MASTER sees the connection attempt. It creates an SSHD process, passing the necessary information to it, such as the server key and operating parameters.
2. SSHD performs validation for the user.
3. Assuming the login is successful, SSHD creates a pseudo terminal for the user (an _FTAnn: device). This device is owned by the user attempting to log in.
4. SSHD creates an interactive process on the pseudo terminal, using the username, priority, and privileges of the user who is attempting to log in. If a command was specified, it is executed and the session is terminated.
5. SSH generates the file SSH_LOG: SSHD.LOG for each connection to the SSH server. Many connections result in many log files. Instead of purging the files on a regular basis, use the following DCL command to limit the number of versions:

```
$ SET FILE /VERSION_LIMIT=x SSH_LOG:SSHD.LOG
```

Note! The value for /VERSION_LIMIT must not be smaller than the maximum number of simultaneous SSH sessions anticipated. If the value is smaller, SSH users may be prevented from establishing sessions with the server.

AUTHORIZED_KEYS File Format

The SYS$DISK:[<login_dir>.SSH]AUTHORIZED_KEYS file lists the RSA keys that are permitted for RSA authentication. Each line of the file contains one key (empty lines and lines starting with a '#' are comments and ignored). Each line consists of the following fields, separated by spaces:

Table 4-2 RSA Keys

Key	Description
bits	Is the length of the key in bits.
comment	Not used for anything (but may be convenient for the user to identify the key).
exponent	Is a component used to identify and make up the key.
modulus	Is a component used to identify and make up the key.
options	Optional; its presence is determined by whether the line starts with a number or not (the option field never starts with a number.)

Note! Lines in this file are usually several hundred characters long (because of the size of the RSA key modulus). You do not want to type them in; instead, copy the IDENTITY.PUB file and edit it. The options (if present) consists of comma-separated option specifications. No spaces are permitted, except within double quotes. Option names are case insensitive.

The following option specifications are supported:

Table 4-3 RSA Key File [AUTHORIZED_KEYS]

Option Specification	Description
Allowforwardingport="<port list>"	Can be followed by any number of port numbers, separated by spaces. Remote forwarding is allowed for those ports whose number matches one of the patterns. You can use '*' as a wildcard entry for all ports. You can use these formats '>x', '<x', and 'x_x' to specify greater than, less than, or inclusive port range. By default, all port forwardings are allowed. The quotes (" ") are required. The <> show a list. Do not use the < > in the specification. For example: `allowforwardingport "2,52,2043"`
Allowforwardingto="<hostname and port list>"	Can be followed by any number of hostname and port number patterns, separated by spaces. A port number pattern is separated from a hostname pattern by a colon. For example: hostname:port Forwardings from the client are allowed to those hosts and port pairs whose name and port number match one of the patterns. You can use '*' and '?' as wildcards in the patterns for host names. Normal name servers are used to map the client's host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as the hostname. You can use '*' as a wildcard entry for all ports. You can use these formats '>x', '<x', and 'x_x' to specify greater than, less than, or inclusive port range. By default, all port forwardings are allowed.

Table 4-3 RSA Key File [AUTHORIZED_KEYS] (Continued)

Option Specification	Description
command=”command”	Specifies the command to be executed whenever this key is used for authentication. The user-supplied command (if any) is ignored. You may include a quote in the command by surrounding it with a backslash (\). Use this option to restrict certain RSA keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Notice that the client may specify TCP/IP and/or X11 forwardings unless they are prohibited explicitly.
Denyforwardingport=”<port list>”	Can be followed by any number of port numbers, separated by spaces. Remote forwardings are disallowed for those ports whose number matches one of the patterns. You can use ‘*’ as a wildcard entry for all ports. You can use these formats ‘>x’, ‘<x’, and ‘x_x’ to specify greater than, less than, or inclusive port range.

Table 4-3 RSA Key File [AUTHORIZED_KEYS] (Continued)

Option Specification	Description
Denyforwardingto=”<hostname port list>”	Can be followed by any number of hostname and port number patterns, separated by spaces. A port number pattern is separated from a hostname by a colon. For example: hostname:port number pattern Forwardings from the client are disallowed to those hosts and port pairs whose name and port number match one of the patterns. You can use ‘*’ and ‘?’ as wildcards in the patterns for host names. Normal name servers are used to map the client’s host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as a host name. You can use ‘*’ as a wildcard entry for all ports. You can use these formats ‘>x’, ‘<x’, and ‘x_x’ to specify greater than, less than, or inclusive port range.
from=”pattern-list”	In addition to RSA authentication, specifies that the fully-qualified name of the remote host must be present in the comma-separated list of patterns. You can use ‘*’ and ‘?’ as wildcards. The list may contain patterns negated by prefixing them with ‘!’; if the fully-qualified host name matches a negated pattern, the key is not accepted. This option increases security. RSA authentication by itself does not trust the network or name servers (but the key). However, if somebody steals the key, the key permits login from anywhere in the world. This option makes using a stolen key more difficult because the name servers and/or routers would have to be comprised in addition to just the key.

Table 4-3 RSA Key File [AUTHORIZED_KEYS] (Continued)

Option Specification	Description
idle-timeout=time	Sets the idle timeout limit to a time in seconds (s or nothing after the number), in minutes (m), in hours (h), in days (d), or in weeks (w). If the connection has been idle (all channels) for that time, the process is terminated and the connection is closed.
no-agent-forwarding	Forbids authentication agent forwarding when used for authentication.
no-port-forwarding	Forbids TCP/IP forwarding when used for authentication. Any port forward requests by the client will return an error. For example, this might be used in connection with the command option.
no-X11-forwarding	Forbids X11 forwarding when used for authentication. Any X11 forward requests by the client will return an error.

RSA Key File Examples

```
1024 33 12121...312314325 ylo@foo.bar
from="*.emptybits.com,!rose.flowers.com"

1024 35 23...2334 ylo@niksula
command="dir *.txt",no-port-forwarding

1024 33 23...2323 xxxxx.tazm.com
allowforwardingport="localhost:80"

1024 35 23...2334 www@localhost
```

SSH_KNOWN_HOSTS File Format

The SSH_DIR:SSH_KNOWN_HOSTS and SYS$DISK:[<login_dir>.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 to 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 wildcards). Each pattern is matched against the fully-qualified host names (when authenticating a client) or against the user-supplied name (when authenticating a server). A pattern may be preceded by '!' to indicate negation; if the hostname 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 host key. They can be obtained from SSH_DIR:SSH_HOST_KEY.PUB. The optional comment field continues to the end of the line, and is not used. 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 permissible (but not recommended) to have several lines or different host keys for the same names. This happens 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.

Note! The lines in these files are hundreds of characters long. Instead of typing in the host keys, generate them by a script or by copying SSH_DIR:SSH_HOST_KEY.PUB and adding the host names at the front.

Example

```
closenet,closenet.hut.fi,...,130.233.208.41
1024  37  159...93 closenet.hut.fi
```

Note! HP's TCP/IP services do not use the traditional UNIX rhosts and hosts.equiv files; it uses a proprietary format. Therefore, any information added to HP's files via the "ADD PROXY" command must also be manually added to the SSH_DIR:RHOSTS and SSH_DIR:HOSTS.EQUIV files in order for it to be used by SSH for OpenVMS.

FILES

Table 4-4 SSH Files

File Name	Description
SSH_DIR:HOSTS.EQUIV	Contains host names, one per line. This file is used during .rhosts authentication. Users on those hosts are permitted to log in without a password, provided they have the same username on both machines. The hostname may also be followed by a username. Such users are permitted to log in as any user on the remote machine (except SYSTEM). Additionally, the syntax +@group can be used to specify netgroups. Negated entries start with '-'. If the client host/user is matched in this file, login is permitted provided the client and server usernames are the same. Successful RSA host authentication is required. This file should be world-readable but writable only by SYSTEM. It is never a good idea to use usernames in hosts.equiv. It means the named user(s) can log in as anybody, which includes accounts that own critical programs and directories. Using a username grants the user SYSTEM access. The only valid use for usernames is in negative entries. ***Note!*** This warning also applies to rshell/rlogin.
SSH_DIR:SHOSTS.EQUIV	Processed as `SSH_DIR:HOSTS.EQUIV`. May be useful in environments that want to run both rshell/rlogin and ssh.

Table 4-4 SSH Files (Continued)

File Name	Description
SSH_DIR:SSH_HOST_KEY	Contains the private part of the host key. This file does not exist when SSH for OpenVMS is installed. The SSH server starts only with this file. This file must be created manually using the command: `$ MULTINET SSHKEYGEN/SSH1/HOST`. This file should be owned by SYSTEM, readable only by SYSTEM, and not accessible to others. To create a host key with a name that is different than what SSHKEYGEN creates, do one of the following: • Generate with MULTINET SSHKEYGEN/SSH1/HOST and simply rename the file(s). • Generate without the /HOST switch and then name the file(s) whatever you want. By default, the logical name SSH_DIR points to the directory where common SSH files are kept, such as the SSH executables.
SSH_DIR:SSH_HOST_KEY.PUB	Contains the public part of the host key. This file should be world-readable but writable only by SYSTEM. Its contents should match the private part. This file is not used for anything; it is only provided for the convenience of the user so its contents can be copied to known hosts files.
SSH_DIR:SSH_KNOWN_HOSTS SYS$DISK:[<login_dir>.SSH]KNOWN_HOSTS	Checks the public key of the host. These files are consulted when using rhosts with RSA host authentication. The key must be listed in one of these files to be accepted. (The client uses the same files to verify that the remote host is the one you intended to connect.) These files should be writable only by SYSTEM (the owner). `SSH_DIR:SSH_KNOWN_HOSTS` should be world-readable, and [.SSH]KNOWN_HOSTS can, but need not be, world-readable.

Table 4-4 SSH Files (Continued)

File Name	Description

SSH_DIR:SSH_RANDOM_SEED	Contains a seed for the random number generator. This file should only be accessible by system. The file SSH_RANDOM_SEED. has the potential for increasing its number of versions. SSH_RANDOM_SEED. is created in the `SSH_DIR:` directory as well as in individual user accounts in the `SYS$LOGIN:[.SSH]` directory.
	
	This DCL command limits the number of versions of this file in the SSH_DIR directory:
	
	`$SET FILE VERSION_LIMIT=x SSH_DIR:SSH_RANDOM_SEED.`
	
	This DCL command limits the number of versions of this file in the user's `SYS$LOGIN:[.SSH]`directory.
	
	`$SET FILE /VERSION_LIMIT=x - SYS$LOGIN:[.SSH]SSH_RANDOM_SEED.`
	
	or
	
	`$CREATE /DIRECTORY /VERSION_LIMIT=x - SYS$LOGIN:[.SSH]SSH_RANDOM_SEED.`

SSH_DIR:SSHD_CONFIG	Contains configuration data for SSHD. This file should be writable by system only, but it is recommended (though not necessary) that it be world-readable.
SYS$DISK:[<login_dir>.SSH] AUTHORIZED_KEYS	Lists the RSA keys that can be used to log into the user's account. This file must be readable by system (which may on some machines imply it being world-readable). It is recommended that it not be accessible by others. The format of this file is described above.
SYS$DISK:[<login_dir>.SSH] SHOSTS	Permits access using SSH only. For SSH, this file is the same as for .rhosts. However, this file is not used by rlogin and rshell daemon.

Table 4-4 SSH Files (Continued)

File Name	Description
SYS$LOGIN: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 a password. The same file is used by rlogin and rshell. SSH differs from rlogin and rshell in that it requires RSA host authentication in addition to validating the hostname retrieved from domain name servers. The file must be writable only by the user. It is recommended that it not be accessible by others. It is possible to use netgroups in the file. Either host or username may be of the form +@groupname to specify all hosts or all users in the group.

SSH Logicals

These logicals are used with the SSH server in the system logical name table.

Table 4-5 SSH Logicals

Logical	Description
SSH_DIR	Points to the directory where system-specific configuration information is kept, including the host key files and the system-specific server configuration file.
SSH_EXE	Points to the directory where common SSH files are kept, such as the SSH executables.
SSH_LOG	Points to the directory where the log files are kept. Normally, this is `MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.LOG]`
SSH_MAX_SESSIONS	Set this to the maximum number of concurrent SSH sessions you want to allow on the server system. If SSH_MAX_SESSIONS is not defined, the default is 9999. Setting SSH_MAX_SESSIONS to zero (0) will cause an error. The value must be between 1 and 9999. It is defined through the configuration procedure.

Table 4-5 SSH Logicals (Continued)

Logical	Description
SSH_TERM_MBX	Mailbox used by SSHD_MASTER to receive termination messages from SSHD daemon processes. **Do not change this logical name.** This is created by the SSHD_MASTER process.
MULTINET_SSH_ALLOW_EXPIRED_PW	Allows logging in to an account when the account's password has expired due to pwdlifetime elapsing. This applies to all users and circumvents normal VMS expired-password checking, and therefore should be used with caution. An entry is made into the SSH_LOG:SSHD.LOG file when access is allowed using this logical name.
MULTINET_SSH_ALLOW_PREEXPIRED_PW	Allows logging in to an account when the password has been pre-expired. This applies to all users and circumvents normal VMS expired-password checking, and therefore should be used with caution. An entry is made into the SSH_LOG:SSHD.LOG file when access is allowed using this logical name.
MULTINET_SSH_KEYGEN_MIN_PW_LEN	Defines the minimum passphrase length when one is to be set in SSHKEYGEN. If not defined, defaults to zero.
MULTINET_SSH_PARAMETERS_*n*	These parameters are used to start SSHD_MASTER. They are parameters set by @MULTINET:CNFSSH CONFIGURE.
MULTINET_SSH_USE_SYSGEN_LGI	If defined, causes SSHD to use the VMS SYSGEN value of LGI_PWD_TMO to set the login grace time, overriding anything specified in the command line or the configuration file.

SSH daemon Files

These files are used by or created by SSH when you log into a daemon. These files are not to be altered in any way.

Table 4-6 SSH daemon Files

File Name	Description
SSHD_MASTER.LOG	This log file is created by SSHD_MASTER.
SSHD.LOG	This log file is created by each SSHD daemon.

Chapter 5

Configuring the Secure Shell (SSH) V2 Server

This chapter describes how to configure and maintain the SSH for OpenVMS SSH V2 server.

This is the server side of the SSH software that allows secure interactive connections from other computers in the manner of rlogin/rshell/telnet. The SSH server has been developed to discriminate between SSH v1 and SSH v2 protocols, so the two protocols can coexist simultaneously on the same system.

SSH1 and SSH2 Differences

SSH1 and SSH2 are different, and incompatible, protocols. The SSH1 implementation is based on the V1.5 protocol and 1.3.7 F-Secure code base, and the SSH2 implementation is based on the V2 protocol and the F-Secure 3.1.0 code base. While SSH2 is generally regarded to be more secure than SSH1, both protocols are offered by the SSH for OpenVMS server, and although they are incompatible, they may exist simultaneously on an SSH for OpenVMS system. The server front-end identifies what protocol a client desires to use, and will create an appropriate server for that client.

Restrictions:

When using SSH to connect to a VMS server, if the VMS account is set up with a secondary password, SSH does not prompt the user for the secondary password. If the VMS primary password entered is valid, the user is logged in, bypassing the secondary password.

When using SSH to execute single commands (in the same manner as RSHELL), some keystrokes like **CTRL/Y** are ignored. In addition, some interactive programs such as HELP may not function as expected. This is a restriction of SSH. If this behavior poses a problem, log into the remote system using SSH in interactive mode to execute the program.

Understanding the SSH for OpenVMS SSH Server

Secure Shell daemon (SSHD) is the daemon program for SSH that listens for connections from clients. The server program replaces rshell and telnet programs. The server/client programs provide secure encrypted communications between two untrusted hosts over an insecure network. A new daemon is created for each incoming connection. These daemons handle key exchange, encryption, authentication, command execution, and data exchange.

Servers and Clients

An SSH for OpenVMS server is an OpenVMS system that acts as a host for executing interactive commands or for conducting an interactive session. The server software consists of two pieces of software (for future reference, “SSHD” will refer to both SSHD_MASTER and SSHD, unless otherwise specified):

- SSHD_MASTER, recognizes the differences between SSH v1 and SSH v2 and starts the appropriate server. If the request is for SSH v1, then the SSH v1 server is run; if the request is for SSH v2, then the SSH v2 server is run.
- SSHD, a copy of which is spawned for each connection instance. SSHD handles all the interaction with the SSH client.

A client is any system that accesses the server. A client program (SSH) is provided with SSH for OpenVMS, but any SSH client that uses SSH version 2 protocol may be used to access the server. Examples of such programs are SSH for OpenVMS, MultiNet SSH2 for OpenVMS, TCPware for OpenVMS; SecureCRT®, and F-Secure SSH Client for Windows®, MacSSH for Macintosh® systems, and other SSH programs on UNIX-based systems.

Each host has a key using DSA encryption and is usually 1024 bits long (although, the user may create a different-sized key, if desired). The same key may be used on multiple machines. For example, each machine in a VMS cluster could use the same key.

When a client connects to the SSHD daemon:

- The client and server together, using the Diffie-Hellman key-exchange method, determine a 256-bit random number to use as the "session key". This key is used to encrypt all further communications in the session.

  Note that this key may be renegotiated between the client and the server on a periodic basis by including the *RekeyIntervalSeconds* keyword in the server configuration file (SSH2_DIR:SSHD2_CONFIG). This is desirable because during long sessions, the more data that is exchanged using the same encryption key, the more likely it is that an attacker who is watching the encrypted traffic could deduce the session key.
- The server informs the client which encryption methods it supports. Currently, AES-128 (the default), Twofish, Blowfish, CAST-128, DES, 3DES, and ARCFOUR are supported by the SSH for OpenVMS system.
- The client selects the encryption algorithm from those offered by the server.
- The client and the server then enter a user authentication dialog. The server informs the client which authentication methods it supports, and the client then attempts to authenticate the user by using some or all of the authentication methods. The following authentication algorithms are

supported:

- public-key (DSA keys)
- hostbased
- password

System security is not improved unless the RLOGIN, RSHELL, and TELNET services are disabled.

If the client authenticates itself successfully, a dialog is entered for preparing the session. At this time the client may request things like:

- forwarding X11 connections
- forwarding TCP/IP connections
- forwarding the authentication agent connection over the secure channel

Finally, the client either requests an interactive session or execution of a command. The client and the server enter session mode. In this mode, either the client or the server may send data at any time, and such data is forwarded to/from the virtual terminal or command on the server side, and the user terminal in 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.

Break-In and Intrusion Detection

Care must be exercised when configuring the SSH clients and server to minimize problems due to intrusion records created by OpenVMS security auditing. The SSH user should consult the system manager to determine the authentication methods offered by the SSH server. The client should then be configured to not attempt any authentication method that is not offered by the server.

If a client attempts authentication methods not offered by the server, the OpenVMS security auditing system may log several intrusion records for each attempt to create a session to that server. The result being that the user could be locked out and prevented from accessing the server system without intervention from the server's system manager.

The authentication methods to be offered by the server are determined by the configuration keywords *AllowedAuthentications* and *RequiredAuthentications*. The number of intrusion records to be logged for any attempted SSH session is determined by the *StrictIntrusionLogging* configuration keyword.

When StrictIntrusionLogging is set to **YES** (the default), each method that is tried and fails causes an intrusion record to be logged. The following rules apply:

- When *HostBased* or *PublicKey* authentications are attempted and fail, one intrusion record is logged for each failed method.
- When password authentication is attempted, one intrusion record is logged for each failed password.

Example 1:

The server is set up to allow *HostBased* and password authentication; also, up to three password attempts are allowed. If all methods fail, four intrusion records are logged:

1 for the failed HostBased
3 for the failed password attempts, one per attempt

When *StrictIntrusionLogging* is set to **NO**, it has the effect of relaxing the number of intrusions logged. Overall failure of all authentication methods simply counts as a single failure, except for password authentication. The following rules apply:

- When password authentication is attempted, one intrusion record is logged for each failed password.
- When any of *HostBased* or *PublicKey* authentication fails, and password authentication is not attempted, exactly one intrusion record is logged, as opposed to one for each failed method.
- When any of *HostBased* or *PublicKey* authentication fails, but password authentication is attempted and succeeds, the only intrusion record(s) logged is one for each failed password attempt.

Example 2:

The server is set up to allow *HostBased* and password authentication; also, up to three password attempts are allowed. If all methods fail, three intrusion records are logged:

0 for the failed HostBased
3 for the failed password attempts, one per attempt

Example 3:

The server is set up to allow *HostBased* and password authentication; also, up to three password attempts are allowed. *HostBased* and RSA fail, but password authentication is successful after 1 failed password. Therefore, one intrusion record is logged:

0 for the failed HostBased
1 for the failed password attempt

Example 4:

The server is set up to allow *HostBased* and *PublicKey* authentication, but not password authentication. If all methods fail, one intrusion record is logged.

Example 5:

The server is set up to allow *HostBased* and *PublicKey* authentication, but not password authentication. *HostBased* authentication fails, but PublicKey succeeds. No intrusion records are logged.

Configuring SSHD Master

The SSHD Master is configured via CNFSSH. See Chapter 3 of the *SSH for OpenVMS Administration and User's Guide* for details on using CNFSSH to configure SSH.

Note! The only supported methods for starting SSH are to use the **@SYS$STARTUP:PSCSSH$STARTUP** command if SSH isn't running, or to use the **SSHCTRL RESTART** command if SSH is currently running.

SSH2 Configuration File

SSHD reads configuration data from its configuration file. By default, this file is SSH2_DIR:SSHD2_CONFIG. However, it may be modified by setting the *ssh2-config-file* parameter with CNFSSH. The file contains keyword value pairs, one per line. Lines starting with # and empty lines are interpreted as comments. The following keywords are possible. Keywords are case insensitive.

Note! HP's TCP/IP services do not use the traditional UNIX rhosts and hosts.equiv files; it uses a proprietary format. Therefore, any information added to HP's files via the "ADD PROXY" command must also be manually added to the SSH_DIR:RHOSTS and SSH_DIR:HOSTS.EQUIV files in order for it to be used by SSH for OpenVMS.

Table 5-1 SSH2 Configuration File Keywords [SSHD2_CONFIG]

Keyword	Value	Default	Description
AllowedAuthentications	List	Publickey, Password	Permitted techniques
AllowGroups	List		Access control by UAF rightslist entries
AllowHosts	Host list		Access control by hostname
AllowShosts	Host list		Access control by hostname
AllowTcpForwarding	Y/N	Y	Enable TCP port forwarding
AllowTcpForwardingForUsers	User list		Per-User forwarding
AllowTcpForwardingForGroups	Rights list		Per-Rightslist ID forwarding
AllowUsers	User list		Access control by username

Table 5-1 SSH2 Configuration File Keywords [SSHD2_CONFIG] (Continued)

Keyword	Value	Default	Description
AllowX11Forwarding	Y/N	Y	Enable X11 forwarding
AuthorizationFile	Filename	Authorization	Authorization file for publickey authentication.
BannerMessageFile	Filename	SYS$ANNOUNCE	Message sent to the client before authentication begins
CheckMail	Y/N	Y	Display information about new mail messages when logging in
Ciphers	Cipher list		Encryption ciphers offered
ClientHostnameDNS			Must match the one in DNS
DenyGroups	Rights list		Deny access for UAF rightslist identifiers
DenyHosts	Host list		Deny access for hosts
DenySHosts	Host list		Deny access for hosts
DenyTcpForwardingForUsers	User list		Forbid forwarding for listed users
DenyTcpForwardingForGroups	Rights list		Forbid forwarding for listed rightslist names
DenyUsers	User list		Access control by username
FascistLogging	Y/N	Y	Verbose logging

Table 5-1 SSH2 Configuration File Keywords [SSHD2_CONFIG] (Continued)

Keyword	Value	Default	Description
ForwardAgent	Y/N	Y	Enable agent forwarding
ForwardX11	Y/N	Y	Enable X11 forwarding
HostbasedAuthForceClient HostnameDNSMatch	Y/N	N	Host name given by client
Hostkeyfile	Filename	Hostkey	Host key filename
IdentityFile	Filename	Identification	Identity filename
IdleTimeout	Time	0 = none	Set idle timeout (in seconds)
IgnoreRhosts	Y/N	N	Ignore local rhosts
IgnoreRootRhosts	Y/N	Y	Ignore system rhosts
KeepAlive	Y/N	Y	Send keepalives
ListenAddress	IP address	0.0.0.0	Listen on given interface
Macs	Algorithm		Select MAC (Message Authentication Code) algorithm
MaxBroadcastsPerSecond	#broadcasts	0	Listen for UDP broadcasts
NoDelay	Y/N	N	Enable Nagel Algorithm
PasswordGuesses	#guesses	3	Limit number of password tries to specified number
PermitEmptyPasswords	Y/N	N	Permit empty (blank) passwords
PermitRootLogin	Y/N	N	SYSTEM can log in

Table 5-1 SSH2 Configuration File Keywords [SSHD2_CONFIG] (Continued)

Keyword	Value	Default	Description
PrintMotd	Y/N	Y	Display SYS$WELCOME when logging in
PublicHostKeyFile	Filename	Hostkey.pub	Host key file location
QuietMode	Y/N	N	Quiet mode
RandomSeedFile	Filename	Random_seed	Random seed file
RekeyIntervalSeconds	#seconds	3600	Frequency of rekeying
RequiredAuthentication	Authentication list		Overrides Allowed Authentications client must support
RequireReverseMapping	Y/N	N	Remote IP address must map to hostname
RSAAuthentication	Y/N	Y	Enable RSA authentication
StrictIntrusionLogging	Y/N	Y	Determine how intrusion records are created by failed authentication attempts
StrictModes	Y/N	N	Strict checking for directory and file protection
SyslogFacility	Syslog level	“AUTH”	Syslog log facility
UserConfigDirectory	Directory	SYS$LOGIN:	Location of user SSH2 directories
UserKnownHosts	Y/N	Y	Respect user [.ssh2] known hosts keys
VerboseMode	Y/N	N	Verbose mode

Table 5-1 SSH2 Configuration File Keywords [SSHD2_CONFIG] (Continued)

Keyword	Value	Default	Description
X11Forwarding	Y/N	Y	Enable X11 forwarding

The keywords MACs and Ciphers have discrete values, plus there are values that actually denote a grouping of 2 or more of the discrete values. Each of these values may be put in the configuration file (SSH2_DIR:SSHD2_CONFIG).

Table 5-2 MAC and Cipher Discrete Values

MACs	discrete values: aes, aes128-cbc, aes256-cbc, aes192-cbc, aes128-cbc, hmac-sha1, hmac-sha1-96, hmac-md5, hmac-md5-96, hmac-ripemd160, hmac-ripemd160-96, sha1-8, sha1, md5-8, md5, ripemd160-8, ripemd160, none
	group ANYMAC consists of: hmac-sha1, hmac-sha1-96, hmac-md5, hmac-md5-96, hmac-ripemd160, hmac-ripemd160-96, sha1-8, sha1, md5-8, md5, ripemd160-8, ripemd160
	group ANY consists of: hmac-sha1, hmac-sha1-96, hmac-md5, hmac-md5-96, hmac-ripemd160, hmac-ripemd160-96, sha1-8, sha1, md5-8, md5, ripemd160-8, ripemd160, none
	group ANYSTD consists of: hmac-md5, hmac-md5-96, hmac-sha1, hmac-sha1-96, none
	group ANYSTDMAC consists of: hmac-md5, hmac-md5-96, hmac-sha1, hmac-sha1-96
Ciphers	discrete values: aes, aes256-cbc, aes192-cbc, aes128-cbc, des, 3des, twofish, blowfish, cast, 3des-cbc, blowfish-cbc, twofish-cbc, arcfour, cast128-cbc, 3des-ecb, 3des-cfb, 3des-ofb, cast128-ecb, cast128-cfb, cast128-ofb, cast128-12-ecb, cast128-12-cbc, cast128-12-cfb, cast128-12-ofb, blowfish-ecb, blowfish-cfb, blowfish-ofb, des-ecb, des-cbc, des-cfb, des-ofb, twofish-ecb, twofish-cfb, twofish-ofb, none
	group ANYSTDCIPHER consists of: aes, aes128-cbc, 3des-cbc, cast128-cbc, blowfish-cbc, twofish-cbc, arcfour, 3des, twofish, blowfish, cast

Table 5-2 MAC and Cipher Discrete Values (Continued)

	group ANY consists of: aes, aes256-cbc, aes192-cbc, aes128-cbc, des, 3des, twofish, blowfish, cast, 3des-cbc, blowfish-cbc, twofish-cbc, arcfour, cast128-cbc, 3des-ecb, 3des-cfb, 3des-ofb, cast128-ecb, cast128-cfb, cast128-ofb, cast128-12-ecb, cast128-12-cbc, cast128-12-cfb, cast128-12-ofb, blowfish-ecb, blowfish-cfb, blowfish-ofb, des-ecb, des-cbc, des-cfb, des-ofb, twofish-ecb, twofish-cfb, twofish-ofb, none
	group ANYCIPHER consists of: aes, aes256-cbc, aes192-cbc, aes128-cbc, des, 3des, twofish, blowfish, cast, 3des-cbc, blowfish-cbc, twofish-cbc, arcfour, cast128-cbc, 3des-ecb, 3des-cfb, 3des-ofb, cast128-ecb, cast128-cfb, cast128-ofb, cast128-12-ecb, cast128-12-cbc, cast128-12-cfb, cast128-12-ofb, blowfish-ecb, blowfish-cfb, blowfish-ofb, des-ecb, des-cbc, des-cfb, des-ofb, twofish-ecb, twofish-cfb, twofish-ofb
	group ANYSTD consists of: aes, aes256-cbc, aes192-cbc, aes128-cbc, 3des-cbc, cast128-cbc, blowfish-cbc, twofish-cbc, arcfour, des, 3des, twofish, blowfish, cast, none

A discrete value or a group identifier may be used with MACS and CIPHERS. For example, in the configuration file, the following examples could be used:

```
Ciphers     ANYCIPHER
Ciphers      cast, des, twofish
MACs        ANYMAC
MACs        hmac-sha1
```

Aiases may be used for some standard ciphers:

Alias	**Value**
aes	aes128-cbc
des	des-cbc
3des	3des-cbc
cast	cast128-cbc
twofish	twofish-cbc
blowfish	blowfish-cbc

Alias	Value
arcfour	arcfour

Starting the SSH Server for the First Time

Follow these instructions to configure the SSH server. If SSH isn't currently running, you must define the MULTINET logicals by using:

```
$ @SYS$STARTUP:PSCSSH$STARTUP LOGICALS
```

1. Use CNFSSH to enable the SSH2 server. For more information, see Chapter 3 of the *SSH for OpenVMS Administration and User's Guide*.
2. Use SSHKEYGEN /SSH2/HOST to generate an SSH2 key and to create the server key in the MULTINET_SSH2_HOSTKEY_DIR directory if it has not previously been created as part of the CNFSSH configuration:

```
$ DEFINE MULTINET_SSH_HOSTKEY_DIR -
_$ MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2.HOSTKEYS]

$ MULTINET SSHKEYGEN /SSH2/HOST
Generating 1024-bit dsa key pair
  8 .oOo.oOoo.oO
Key generated.
1024-bit dsa, lillies@flower.plants.com, Mon Aug 06 2002 09:19:47
Private key saved to multinet_ssh2_hostkey_dir:hostkey.
Public key saved to multinet_ssh2_hostkey_dir:hostkey.pub
```

3. Edit the configuration file at SSH2_DIR:SSHD2_CONFIG (if you wish to change the default settings). This default configuration is the same as contained in the file MULTINET:SSHD2_CONFIG.TEMPLATE

Note! As delivered, the template file provides a reasonably secure SSH environment. However, Process Software recommends this file be examined and modified appropriately to reflect the security policies of your organization.

4. Start SSH. This creates the SSH server process and defines the SSH logical names.

```
$ @SYS$STARTUP:PSCSSH$STARTUP
$ SHOW PROCESS "SSHD Master"
```

```
7-JUN-2002 09:03:06.42  User: SYSTEM      Process ID:   00000057
                        Node: PANTHR      Process name: "SSHD Master"

Terminal:
User Identifier:    [SYSTEM]
Base priority:      4
Default file spec:  Not available
Number of Kthreads: 1

Devices allocated:  BG1:
                    BG2:
```

```
$ SHOW LOGICAL/SYSTEM *SSH*

  "MULTINET_SSH2_HOSTKEY_DIR" ="MULTINET_SPECIFIC_ROOT:
  [MULTINET.PSCSSH.SSH2.HOSTKEYS]"
  "MULTINET_SSH2_KNOWNHOSTS_DIR"=
  "MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2.KNOWNHOSTS]"
  "MULTINET_SSH_ALLOW_EXPIRED_PW"="1"
  "MULTINET_SSH_ALLOW_PREEXPIRED_PW"="1"
  "MULTINET_SSH_DISPLAY_SYS$ANNOUNCE"="1"
  "MULTINET_SSH_ENABLE_SSH1_CONNECTIONS"="1"
  "MULTINET_SSH_ENABLE_SSH2_CONNECTIONS"="1"
  "MULTINET_SSH_LOG_MBX" = "MBA37"
  "MULTINET_SSH_PARAMETERS_0"="/BITS=768/VERBOSE/QUIET/PORT=22"
  "MULTINET_SSH_PARAMETERS_1"="/KEY_GEN_TIME=3600"
  "MULTINET_SSH_PARAMETERS_2"=""
  "MULTINET_SSH_PARAMETERS_3" =""
  "SSH2_DIR"=MULTINET__SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2]"
  "SSH_DIR"="MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH]"
  "SSH_EXE"= MULTINET_COMMON_ROOT:[MULTINET.PSCSSH]"
  "SSH_LOG"= MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.LOG"
  "SSH_MAX_SESSIONS"="100"
  "SSH_TERM_MBX"="MBA36:"
```

Changing SSH2 Configuration File After Enabling SSH2

```
$ SSHCTRL RESTART
```

Note! When issuing the "RESTART" command for SSH, all active SSH server sessions are terminated. Active client sessions are not affected.

Connection and Login Process

To create a session, SSHD does the following:

1. SSHD_MASTER sees the connection attempt. It creates an SSHD process, passing the operating parameters to it. SSHD performs validation for the user.
2. Assuming the login is successful, SSHD creates a pseudo terminal for the user (an FTAnn: device). This device is owned by the user attempting to log in.
3. SSHD creates an interactive process on the pseudo terminal, using the username, priority, and privileges of the user who is attempting to log in. If a command was specified, it is executed and the session is terminated.
4. SSH generates the file SSHD.LOG for each connection to the SSH server. Many connections result in many log files. Instead of purging the files on a regular basis, use the following DCL command to limit the number of versions:

```
$ SET FILE /VERSION_LIMIT=x SSH_LOG:SSHD.LOG
```

Note! The value for /VERSION_LIMIT must not be smaller than the maximum number of simultaneous SSH sessions anticipated. If the value is smaller, SSH users may be prevented from establishing sessions with the server.

SSH Files

Note! HP's TCP/IP services do not use the traditional UNIX rhosts and hosts.equiv files; it uses a proprietary format. Therefore, any information added to HP's files via the "ADD PROXY" command must also be manually added to the SSH_DIR:RHOSTS and SSH_DIR:HOSTS.EQUIV files in order for it to be used by SSH for OpenVMS.

The following table provides descriptions of the various SSH files:

Table 5-3 SSH Files

File	Description
SSH_DIR:HOSTS.EQUIV	Contains host names, one per line. This file is used during rhosts authentication. Users on those hosts are permitted to log in without a password, provided they have the same username on both machines. The hostname may also be followed by a username. Such users are permitted to log in as any user on the remote machine (except SYSTEM). Additionally, the syntax +@group can be used to specify netgroups. Negated entries start with dash (-). If the client host/user is matched in this file, login is permitted, provided the client and server usernames are the same. Successful RSA host authentication is required. This file should be world-readable but writable only by SYSTEM. It is never a good idea to use usernames in hosts.equiv. It means the named user(s) can log in as anybody, which includes accounts that own critical programs and directories. Using a username grants the user SYSTEM access. The only valid use for usernames is in negative entries. ***Note!*** This warning also applies to rshell/rlogin.
SSH_DIR:SHOSTS:EQUIV	Processed as `SSH_DIR:HOSTS.EQUIV`. May be useful in environments that want to run both rshell/rlogin and ssh.

Table 5-3 SSH Files

File	Description
MULTINET_SSH2_HOSTKEY_DIR:HOSTKEY	Contains the private part of the host key. This file does not exist when SSH for OpenVMS is first installed. The SSH server requires this file to exist to start. This file must be created manually using the command: **$ MULTINET SSHKEYGEN /SSH2 / HOST** This file should be owned by SYSTEM, readable only by SYSTEM, and not accessible to others. To create a host key with a name that is different than what SSHKEYGEN creates, do one of the following: • Generate with SSHKEYGEN /SSH2 / HOST and simply rename the file(s). • Generate without the /HOST switch and then name the file(s) whatever you want. By default, the logical name SSH2_DIR points to the MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2] directory. Refer to Chapter 6 of the *SSH for OpenVMS Administration and User's Guide*, for more details about SSHKEYGEN.
MULTINET_SSH2_HOSTKEY_DIR:HOSTKEY.PUB	Contains the public part of the host key. This file should be world-readable but writable only by SYSTEM. Its contents should match the private part. This file is not used for anything; it is only provided for the convenience of the user so its contents can be copied to known hosts files
SSH2_DIR:RANDOM_SEED	Seeds the random number generator. This file should not be readable by anyone but SYSTEM.
SYS$LOGIN:[.SSH2]RANDOM_SEED	Seeds the random number generator. This file should not be readable by anyone but the user.

Table 5-3 SSH Files

File	Description
SYS$LOGIN:RHOSTS	This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without a password. The same file is used by rlogin and rshell. SSH2 differs from rlogin and rshell in that it requires RSA host authentication in addition to validating the hostname retrieved from domain name servers (unless compiled with the -with-rhosts configuration option). The file must be writable only by the user. It is recommended that it not be accessible by others. It is possible to use netgroups in the file. Either host or username may be of the form +@groupname to specify all hosts or all users in the group.
SYS$LOGIN:[.SSH2]AUTHORIZATION	This file contains information on how the server verifies the identity of a user.
SYS$LOGIN:[.SSH2.KNOWNHOSTS]*xxxxyyyy.pub*	These are the public host keys of hosts that a user wants to log in from using "hostbased" authentication (equivalent to the SSH1's "RhostsRSAAuthentication"). Also, a user must set up his/her individual .SHOSTS or .RHOSTS file. If the username is the same in both hosts, it is adequate to put the public host key in the MULTINET_SSH2_KNOWNHOSTS_DIR directory and add the host's name to the system-wide SHOSTS.EQUIV or RHOSTS.EQUIV file. *xxxx* is the hostname (FQDN) and *yyyy* denotes the public key algorithm of the key ("ssh-dss" or "ssh-rsa"). key ("ssh-dss" or "ssh-rsa"). For example flower.plants.com's host key algorithm is "ssh-dss". The hostkey would then be "flower_plants_com_ssh-dss.pub" in the [.SSH2.KNOWNHOSTS] directory.

SSH2 AUTHORIZATION File Format

The Authorization file contains information on how the server verifies the identity of a user. This file has the same general syntax as the SSH2 configuration files, as shown in the following table.

Table 5-4 SSH2 AUTHORIZATION Keywords

Keyword	Description
KEY	The filename of a public key in the [.SSH2] directory in the user's SYS$LOGIN directory. This key is used for identification when contacting the host. If there are multiple KEY lines, all are acceptable for login.
COMMAND	This keyword, if used, must follow the KEY keyword above. This is used to specify a "forced command" that executes on the server side instead of anything else when the user is authenticated. This option might be useful for restricting certain public keys to perform certain operations.

SSH2 Logicals

These logicals are used with the SSH server in the system logical name table.

Table 5-5 SSH2 Logicals

Logical	Description
SSH_DIR	Points to the directory where the master server log file is kept. Normally, this is `MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH]`.
SSH_EXE	Points to the directory where common SSH files are kept. Normally, this is `MULTINET_COMMON_ROOT:[MULTINET.PSCSSH]`.
SSH_LOG	Points to the directory where the log files are kept. Normally, this is `MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.LOG]`

Table 5-5 SSH2 Logicals (Continued)

Logical	Description
MULTINET_SSH_MAX_SESSIONS	Set this to the maximum number of concurrent SSH sessions you want to allow on the server system. If `MULTINET_SSH_MAX_SESSIONS` is not defined, the default is 1000. Setting `MULTINET_SSH_MAX_SESSIONS` to zero (0) will cause an error. The value must be between 1 and 1000.
SSH_TERM_MBX	Mailbox used by SSHD_MASTER to receive termination messages from SSHD daemon processes. **Do not change this logical name.** This is created by the SSHD_MASTER process.
MULTINET_SSH_PARAMETERS_*n*	These values are set by MultiNet and must not be modified by the user.
MULTINET_SSH_ENABLE_SSH2_CONNECTIONS	Enables SSHD Master to accept SSH V2 sessions.
MULTINET_SSH2_HOSTKEY_DIR	Directory containing the host keys for the SSH V2 server. Normally set to `MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2.HOSTKEYS]`
MULTINET_SSH2_KNOWNHOSTS_DIR	Directory containing client's public host keys used for hostbased authentication. Normally set to MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2.KNOWNHOSTS]
SSH2_DIR	Contains all SSH V2-specific files, such as configuration files. Normally set to `MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2]`.

SSH daemon Files

These files are used by or created by SSH when you log into a daemon. These files are not to be altered in any way.

Table 5-6 SSH daemon Files

File	Description
SSH_LOG:SSHD.LOG	This log file is created by each SSHD daemon.
SSHD_MASTER.LOG	This log file is created by SSHD_MASTER.

Chapter 6

Accessing Remote Systems with the Secure Shell (SSH) Utilities

The SSH implementation for SSH for OpenVMS provides the client software for allowing secure interactive connections to other computers in the manner of rlogin/rshell/telnet.

The following topics describe how to configure, maintain, and use the following SSH for OpenVMS clients:

- Secure Shell Client (remote login program)
- SSHKEYGEN
- SSHAgent (authentication agent)
- SSHADD

SSH Protocol Support

The SSH client software supports both the SSH1 and SSH2 protocols. SSH1 and SSH2 are different, and incompatible protocols. The SSH1 implementation is based on the V1.5 protocol and 1.3.7 F-Secure code base, and the SSH2 implementation is based on the V2 protocol and the F-Secure 3.1.0 code base. While SSH2 is generally regarded to be more secure than SSH1, both protocols are offered by SSH for OpenVMS, and although they are incompatible, they may exist simultaneously on server systems, including SSH for OpenVMS servers. The SSH client identifies the protocol(s) offered by any given server. If both SSH2 and SSH1 protocols are offered, the client will always use SSH2. Otherwise, the client will use the correct protocol based on the server's capability.

Secure Shell Client (remote login program)

SSH (Secure Shell) is a program for logging into and executing commands on a remote system. It replaces rlogin, rsh, and telnet, and provides secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can be forwarded over the secure channel. SSH connects and logs into the specified hostname. The user must prove his/her identity to the remote system using one of several methods.

Initial Server System Authentication

When an initial connection is made from the client system to the server system, a preliminary authentication of the server is made by the client. To accomplish this, the server system sends its public key to the client system.

SSH maintains a directory containing the public keys for all hosts to which it has successfully connected. For each user, this is the [.SSH2.HOSTKEYS] directory off the individual SYS$LOGIN directory[1]. In addition, a system-wide directory of known public keys exists in the system directory pointed to by the logical name MULTINET_SSH2_HOSTKEY_DIR, and this may be populated by the system manager. Both directories are searched as needed when establishing a connection between systems. Any new host public keys are added to the user's HOSTKEYS directory. If a host's identification 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 that could be used to circumvent the encryption. The SSH configuration option *StrictHostKeyChecking* can be used to prevent logins to a system whose host key is not known or has changed.

Hostbased Authentication

Hostbased authentication relies on two things: the existence of the user's system and username in either SSH_DIR:HOSTS.EQUIV or in the individual user's SYS$LOGIN:.RHOSTS or SYS$LOGIN:.SHOSTS file; and the server system having prior knowledge of the client system's public host key.

Note! HP's TCP/IP services do not use the traditional UNIX rhosts and hosts.equiv files; it uses a proprietary format. Therefore, any information added to HP's files via the "ADD PROXY" command must also be manually added to the SSH_DIR:RHOSTS and SSH_DIR:HOSTS.EQUIV files in order for it to be used by SSH for OpenVMS.

- For SSH2

  When a user logs in:

  1 The server checks the SSH_DIR:HOSTS.EQUIV file, and the user's SYS$LOGIN:.RHOSTS and SYS$LOGIN:.SHOSTS files for a match for both the system and username. Wildcards are not permitted.

1. In this chapter, the [.SSH] subdirectory in the user's login directory displays as SYS$LOGIN:[.SSH] [.SSH2] displays as SYS$LOGIN:[.SSH2]

2. The server checks to see if it knows of the client's public host key (SSH2_DIR:HOSTKEY.PUB on VMS client systems) in either the user's SYS$LOGIN:[SSH2.KNOWNHOSTS] directory or in the system-wide directory pointed to by the MULTINET_SSH2_KNOWNHOSTS_DIR logical name. The key file is named <FQDN>_<algorithm>.PUB. For example, if the client system is "foo.bar.com" and its key uses the DSS algorithm, the file that would contain its key on the server would be "FOO_BAR_COM_SSH-DSS.PUB". This key file must exist on the server system before attempting hostbased authentication.
3. If the key file is found by the server, the client sends its digitally-signed public host key to the server. The server will check the signature for validity.

- For SSH1

  This form of authentication alone is not allowed by the server because it is not secure. The second (and primary) authentication method is the RHOSTS or HOSTS.EQUIV method combined with RSA-based host authentication. It means that if the login would be permitted by .RHOSTS, .SHOSTS, SSH_DIR:HOSTS.EQUIV, or SSH_DIR:SHOSTS.EQUIV file, and if the client's host key can be verified (see SYS$LOGIN:[.SSH]KNOWN_HOSTS and SSH_DIR:SSH_KNOWN_HOSTS in the FILES section), only then is login permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing.

Note! To the administrator: SSH_DIR:HOSTS.EQUIV,.RHOSTS, and the rlogin/rshell protocol are inherently insecure and should be disabled if security is desired.

Publickey Authentication

The SSH client supports DSA-based authentication for SSH2 sessions, and RSA-based authentication for SSH1 sessions. 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.

- For SSH1

  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 (SYS$LOGIN:[.SSH]AUTHORIZED_KEYS lists the public keys permitted for log in), and only the user knows the private key.

  When the user logs in:

  1. The SSH client program tells the server the key pair it would like to use for authentication.
  2. The server checks if this key pair is permitted.

     If it is permitted, the server sends the SSH client 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.

3 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.

4 SSH implements the RSA authentication protocol automatically.

The Key Identity files are created with SSHKEYGEN. To create the RSA key pair files with SSH for OpenVMS:

1 Run SSHKEYGEN to create the RSA key pair: IDENTITY and IDENTITY.PUB.

Both of these files are stored in the user's SYS$LOGIN:[.SSH]directory. IDENTITY.; is the private key; IDENTITY.PUB is the public key.

Once you have created your identity files:

1 Transfer the IDENTITY.PUB file to the remote machine.

2 Update the AUTHORIZED_KEYS file on the remote machine by appending the contents of the public key file to the SYS$LOGIN:[.SSH]AUTHORIZED_KEYS file on the remote host. The format of the AUTHORIZED_KEYS file requires that each entry consists of a single long line.

After this, the user can log in 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 *Publickey Authentication* for more information.

- For SSH2

When the user logs in:

1 The client reads possible keys to be used for authentication from its IDENTIFICATION file. Note that this file does not contain the actual keys; rather, it contains the name of the key files.

2 The client sends to the server its list of keys.

3 The server compares each key that it received to see if it can match this key with one of those specified in the AUTHORIZATION file.

4 The server tells the client the key that was accepted. The client then "signs" the key with a digital signature that only the server with the proper key could verify, and sends the signature to the server.

5 The server verifies the signature.

Password authentication

The password is sent to the remote host for checking. The password cannot be seen on the network because all communications are encrypted. When the server accepts the user's identity it either executes the given command or logs into the system and gives the user a normal shell on the remote system. All communication with the remote command or shell will be encrypted automatically.

Expired Passwords

The SSH client supports the changing of expired passwords for SSH2 sessions only. When a password expires that may be changed by the user, the user will be prompted to re-enter the original password, then to enter a new password twice.

The new password is validated against the password history maintained by VMS. It is also validated against the VMS system dictionary of restricted passwords.

Note! If a user is required to use system-generated passwords, either because the GENPWD flag is set in SYSUAF or because the user has exceeded the limits on the number of password history entries, the user will not be allowed to log in via SSH. This is because the protocol does not allow for this VMS-specific feature. In these cases, the user must either log in via Telnet and reset the password that way, or the user must contact the system manager to reset the password.

The SSH v1 protocol does not provide a method for changing an expired VMS password. When an expired password is encountered by the SSH1 server, it will do one of two things.

1. If the logical name MULTINET_SSH_ALLOW_EXPIRED_PW is defined for allowing access for passwords that have exceeded the UAF value for PWDLIFETIME, or if the logical name MULTINET_SSH_ALLOW_PREEXPIRED_PW is defined for allowing access for users that have a pre-expired password, the server will allow the user to log in. In the logical name table LNM$SSH_LOGICALS, the logical name MULTINET_SSH_*pid*_PWDEXP (where *pid* is the process ID for the user process) will be defined. The system manager can look for this logical to be defined, and if so, take action such as executing the DCL SET PASSWORD command.
2. If the appropriate logical is not set as described above, the user will be denied access to the system. In that case, the user must log in interactively via another mechanism such as telnet and change the password, or the system manager must reset the password.

Break-in and Intrusion Detection

Care must be exercised when configuring the client to minimize problems due to intrusion records created by OpenVMS security auditing. The SSH user should consult the system manager to determine the authentication methods offered by the SSH server. Examples of such authentication methods include HostBased, PublicKey, and Password. The client should be configured to not attempt any authentication method that is not offered by the server.

If a client attempts authentication methods not offered by the server, the OpenVMS security auditing system may log several intrusion records for each attempt to create a session to that server. The result being that the user could be locked out and prevented from accessing the server system without intervention from the server's system manager.

Session Termination

The user can disconnect with "~.". All forwarded connections can be listed with "~#". All available escapes can be listed with "~?". A single tilde character can be sent as "~~" (or by following the tilde with a character other than those described above). The escape character must always follow a carriage return to be interpreted as special. The escape character "~" can be changed in configuration files or on the command line.

The session terminates when the command or shell on the remote system exits, or when the user logs out of an interactive session, 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.

X11 Forwarding

With X11 in use, the connection to the X11 display forwards to the remote side any X11 programs started from the interactive session (or command) through the encrypted channel. Also, the connection to the real X server is made from the local system. The user should not set DECW$DISPLAY manually. Forwarding of X11 connections can be configured on the command line or in configuration files.

The DECW$DISPLAY value set by SSH points to the server system with a display number greater than zero. This is normal and happens because SSH creates a "proxy" X server on the server system for forwarding the connections over the encrypted channel.

SSH sets up “fake” Xauthority data on the OpenVMS server, as OpenVMS does not support Xauthority currently. 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 by the real cookie when the connection is opened. The real authentication cookie is never sent to the server system (and no cookies are sent in plain text).

Configuring the SSH Client

The SSH client uses only SSH2 configuration keywords. There are no SSH1-specific configuration keywords for the SSH client.

The SSH client obtains configuration data from the following sources (in this order):

1 Command line options. See Table 6-1 for details.

2 User's configuration file (in the local SYS$LOGIN [.SSH2]SSH2_CONFIG.) directory. See Table 6-2 for details.

3 System-side configuration file (SSH2_DIR:SSH2_CONFIG.) See Table 6-2 for details.

For each parameter, the first obtained value is used. The configuration files contain sections bracketed by "Host" specifications. That section applies only 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.

Note! The qualifiers listed in Table 6 -1 are position dependent. You must place the qualifier(s) immediately after the SSH command. So the correct syntax is `SSH /qualifier node command`.

Table 6 -1 SSH Client Command Options and Qualifiers

Qualifier	Description
/ALLOW_REMOTE_CONNECT	Allow remote hosts to connect local port forwarding ports. The default is only localhost; may connect to locally binded ports.
/CIPHER=(*cipher-1,...,cipher-n*)	Select encryption algorithm(s).
/COMPRESS	Enable compression.
/CONFIG_FILE=*file*	Read an alternative config file.
/DEBUG=*level*	Set debug level.
/ESCAPE_CHARACTER=*char*	Set escape character; “none” = disable (default: ~).
/HELP	Display help text.
/IDENTITY_FILE=*file*	Identity file for public key authentication.
/LOCAL_FORWARD= ([protocol/]*listen-port:host:port,...)*	Causes the given port on the local (client) host to be forwarded to the given host and port on the remote side. The system to which SSH connects acts as the intermediary between the two endpoint systems. Port forwardings can be specified in the configuration file. Only system can forward privileged ports. See the Port Forwarding section for more details.
/LOG_FILE=*logfilename*	Log all terminal activity to the specified log file. Defaults to SSH.LOG if “*logfilename*” is not specified.
/MAC=*(mac-1,...,mac-n)*	Select MAC algorithm(s).
/NO_X11_FORWARDING	Disable X11 connection forwarding.
/OPTION=*(option-1,...option-n)*	Gives options in the format used in the configuration file. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file, and are processed prior to any keywords in the configuration file. For example: /OPTION=(CompressionLevel=6)

Table 6 -1 SSH Client Command Options and Qualifiers (Continued)

Qualifier	Description
/PORT=*port*	Connect to this port on server system. Server must be listening on the same port.
/QUIET	Quiet Mode. Causes all warning and diagnostic messages to be suppressed. Only fatal errors display.
/REMOTE_FORWARD= ([protocol/]*listen-port:host:port,...*)	Forward remote port to local address. These cause ssh to listen for connections on a port, and forward them to the other side by connecting to host port.
/USE_NONPRIV_PORT	Use a non-privileged (>1023) source port.
/USER=*user*	Log in to the server system using this user name.
/VERBOSE	Display verbose debugging messages. Equal to "/DEBUG=2".
/VERSION	Display version number of the client.

Table 6-2 SSH2_CONFIG File Configuration Keywords

Keyword	Value	Default	Description
AllowedAuthentications	List	PublicKey, Password	Permitted techniques, listed in desired order of attempt.
AuthenticationSuccessMsg	Y/N	Y	Print message on successful authentication
AuthorizationFile	Filename	Authorization	Authorization file for publickey authentication
BatchMode	Y/N	N	Don't prompt for any input during session
Ciphers	Cipher list	None	Supported encryption ciphers
ClearAllForwardings	Y/N	N	Ignore any specified forwardings
CompressionLevel	Y/N	N	Enable data compression
ConnectionAttempts	Number of attempts	4	Number of retries by client to connect to the server
DefaultDomain	Domain		Specify domain name

Table 6-2 SSH2_CONFIG File Configuration Keywords (Continued)

Keyword	Value	Default	Description
EscapeChar	Character	“~”	Set escape character (^=ctrl key)
ForwardAgent	Y/N	Y	Enable agent forwarding
ForwardX11	Y/N	Y	Enable X11 forwarding
GatewayPorts	Y/N	N	Gateway locally forwarded ports
Host	Pattern		Begin section for this host
IdentityFile	Filename	Identification	Name of identification file for publickey authentication
KeepAlive	Y/N	Y	Send keepalives
LocalForward	Port, Socket		Local port forwarding
Macs	Algorithm	None	Select MAC (Message Authentication Code) algorithm
NoDelay	Y/N	N	Disable Nagle (TCP_NODELAY)
NumberOfPasswordPrompts	Number	3	Number of times the user is prompted for a password before the connection is dropped
PasswordPrompt	String	“%U’s password:”	Password prompt
PasswordPromptLogin	Y/N	Y	Username for password prompt
Port	Port	22	Server port number
QuietMode	Y/N	Y	Quiet mode - only fatal errors are displayed
RandomSeedFile	Filename	Random_seed	Random seed file

Table 6-2 SSH2_CONFIG File Configuration Keywords (Continued)

Keyword	Value	Default	Description
RekeyIntervalSeconds	Seconds	3600	Number of seconds between doing key exchanges during a session. 0 = disable
RemoteForward	Port, Socket		Remote port forwarding
SendNOOPPackets	Y/N	N	Send NOOP packets through the connection. Used typically to prevent a firewall from closing an interactive session
StrictHostKeyChecking	Y/N/Ask	Y	Behavior on host key mismatch
User	Username		Remote username
VerboseMode	Y/N	N	Verbose mode

Notes Regarding SSH2_CONFIG

The user may specify default configuration options for different destination systems. The format of this within the configuration file is:

```
hostname:
  keyword        value
  keyword        value

hostname2:
   keyword       value
   keyword       value
```

For example:

```
petunia:
  port          17300
  user          dilbert
  host          petunia.flowers.com

rose:
  port          16003
  user          dogbert
  host          rose.flowers.com
  allowedauthentications password
```

```
*.beans.com:
  user            limabean
  keepalive         no
  ciphers         3des,twofish
```

In the preceding example:

- When a user types "$ SSH PETUNIA", the system will connect to port 17300 on petunia.flowers.com, and will use the default username of "dilbert".
- When a user types "$ SSH ROSE", the system will connect to port 16003 on host rose.flowers.com, and will use the default username of "dogbert", and only allow password authentication.
- When a user types "$ SSH <anything>.BEANS.COM", the system will use the default username of "limabean", and will not send keepalives, and will only allow 3DES or TWOFISH encryption.

The user may override defaults specified in configurations. Options that are specified on the command line override any like options in the configuration file. For example, if the user wants to use a username of "catbert" when connecting to host rose instead of the default username of "dogbert", this would be specified as:

```
$ SSH /USER=CATBERT ROSE
```

SSH Client/Server Authentication Configuration Examples

Hostbased Authentication Example

The following is an example of how to set up the SSH client and SSH2 server for Hostbased Authentication:

```
$!
$! First, generate the host key - ONLY if it doesn't exist!
$!
$ multinet sshkeygen /ssh2 /host
Generating 1024-bit dsa key pair
4 oOo.oOo.oOo

Key generated.
1024-bit dsa, myname@myclient.foo.com, Thu JUN 06 2002 13:43:54
Private key saved to multinet_ssh2_hostkey_dir:hostkey.
Public key saved to multinet_ssh2_hostkey_dir:hostkey.pub

$ directory multinet_ssh2_hostkey_dir:hostkey.*

Directory MULTINET_SPECIFIC_ROOT:[MULTINET.PSCSSH.SSH2.HOSTKEYS]
```

```
HOSTKEY.;1          HOSTKEY.PUB;1

Total of 2 files
$!
$! Copy the client system public key to the user directory on the server
$!
$! DECnet must be running before you execute the following commands:
$!
$ copy multinet_ssh2_hostkey_dir:hostkey.pub -
_$ myserv"myname myuser"::[.ssh2.knownhosts]myclient_foo_com_ssh-dss.pub
$!
$! Finally, log into the server system and ensure the
$! SSH_DIR:HOSTS.EQUIV file is correct
$!
$ SET HOST MYSERV

      Welcome to OpenVMS (TM) VAX Operating System, Version V7.3

Username: myname
Password:
      Welcome to OpenVMS VAX V7.3

    Last interactive login on Monday,  3-JUN-2002 17:07
    Last non-interactive login on Monday, 3-JUN-2001 08:30

MYSERV_$ type ssh_dir:hosts.equiv
#
# HOSTS.EQUIV - names of hosts to have default "r" utility access to the local
# system.
#
#   This file should list the full domain-style names.
#
#   This list augments the users' SYS$LOGIN:.RHOSTS file for authentication.
#   Both the .RHOSTS and the HOSTS.EQUIV files are cached by multinet -
#   see the section entitled "RLOGIN and RSHELL Authentication Cache"
#   in the _Administrator's Guide_ for more information on controlling
#   the cache.
#
#   This file is ignored for the users SYSTEM and ROOT. SYSTEM and ROOT
#   must have a SYS$LOGIN:.RHOSTS file if you want to use RSHELL or RLOGIN
#   with them.
#
localhost
myclient.foo.com       myname
MYSERV_$
MYSERV_$ logout
  MYNAME      logged out at  3-JUN-2002 13:46:58.91
%REM-S-END, control returned to node MYCLIENT::
```

Publickey Authentication Example

The following is an example of how to set up the SSH client and SSH2 server for Publickey Authentication:

```
$!
$! First, generate a key tuple
$!
$ multinet sshkeygen /ssh2
Generating 1024-bit dsa key pair
   1 oOo.oOo.oOo.

Key generated.
1024-bit dsa, myname@myclient.foo.com, Thu Jun 06 2001 14:06:10
Passphrase :
Again      :
Private key saved to DISK$USERDISK:[MYNAME.SSH2]id_dsa_1024_a.
Public key saved to DISK$USERDISK:[MYNAME.SSH2]id_dsa_1024_a.pub
$ directory [.ssh2]id*.*/since

Directory DKA0:[MYNAME.SSH2]

ID_DSA_1024_A.;1    ID_DSA_1024_A.PUB;1

Total of 2 files.
$!
$! Now create the IDENTIFICATION. file.  This contains the name of
$! all the keys you wish to use for public-key authentication.
$!
$ set default [.ssh2]
$ copy tt: identification.
  idkey id_dsa_1024_a
  ^Z
$!
$! Copy the key to the user's [.ssh2] directory on the server system
$!
$ copy id_dsa_1024_a.pub myserv"myname mypass"::[.ssh2]
$!
$! Now log into the server system and create the AUTHORIZATION file
$!
$ set host myserv

      Welcome to OpenVMS (TM) VAX Operating System, Version V7.3

Username: myname
Password:
      Welcome to OpenVMS VAX V7.3

    Last interactive login on Tuesday,  4-JUN-2002 13:46
    Last non-interactive login on Tuesday,  4-JUN-2002 13:47
```

```
$ set default [.ssh2]
$ directory [.ssh2]id*.*

Directory DKA0:[MYNAME.SSH2]

ID_DSA_1024_A.PUB;1

Total of 1 file.
$ copy tt: authorization.
key id_dsa_1024_a.pub
^Z
$ logout
  MYNAME      logged out at 4-JUN-2002 14:10:26.16
%REM-S-END, control returned to node MYCLIENT::
```

SSH1 Example

```
   $ ! An example of the procedure of setting up SSH to enable
   $ ! RSA-based authentication.
   $ ! Using SSH client node to connect to an SSH server node.
   $ !
   $ ! On the client node
   $ !
   $ MULTINET SSHKEYGEN /SSH1
   Initializing random number generator...
   Generating p:  ..............................++ (distance 662)
   Generating q:  ................++ (distance 370)
   Computing the keys...
   Testing the keys...
   Key generation complete.
   Enter file in which to save the key
   (DISK$SYS_LOGIN:[MYNAME.ssh]identity.):
   Enter passphrase:
   Enter the same passphrase again:
   Your identification has been saved in
   DISK$SYS_LOGIN:[MYNAME.ssh]identity..
   Your public key is:
   1024 33 13428.........29361 MYNAME@long.hair.com
   Your public key has been saved in DISK$SYS_LOGIN:[MYNAME.ssh]identity.pub
   $ !
   $ ! A TCP/IP stack must be loaded on the remote system.
   $ !
   $  FTP DAISY /USER=MYNAME/PASSWORD=DEMONSOFSTUPIDITY -
   _$ PUT DISK$SYS_LOGIN:[MYNAME.ssh]identity.PUB -
   _$ DISK$SYS_LOGIN:[MYNAME.ssh]identity.PUB
   long.hair.com MultiNet FTP user process V4.4(119)
   Connection opened (Assuming 8-bit connections)
   <daisy.hair.com MultiNet FTP Server Process V4.4(16) at Thu 6-Jun-2002
   3:20PM-EDT
   [Attempting to log in as myname]
```

```
<User MYNAME logged into DISK$SYS_LOGIN:[MYNAME] at Thu 6-JUN-2002 3:21PM-
EDT, job 20e00297.
<VMS Store of DISK$SYS_LOGIN:[MYNAME.SSH]IDENTITY.PUB; started.
<Transfer completed.  395 (8) bytes transferred.
<QUIT command received. Goodbye.
$
$ TELNET DAISY
Trying... Connected to DAISY.HAIR.COM.


        Authorized Users Only (TM) VAX Operating System, Version V7.1

Username: MYNAME
Password:
     Welcome to OpenVMS (TM) VAX Operating System, Version V7.1 on node
DAISY
    Last interactive login on Thursday,  6-JUN-2002 08:07
    Last non-interactive login on Thursday,  6-JUN-2002 15:21
         Logged into DAISY at  6-JUN-2002 15:22:43.68
$ !
$ ! For the first entry into the AUTHORIZED_KEYS file copy
$ ! (or rename) the file [.SSH]IDENTITY.PUB to [.SSH]AUTHORIZED_KEYS.
$ !
$ COPY [.SSH]IDENTITY.PUB [.SSH]AUTHORIZED_KEYS.
$
$ ! FOR SUBSEQUENT ENTRIES use the APPEND command
$ !
$ APPEND [.SSH]IDENTITY.PUB [.SSH]AUTHORIZED_KEYS.
$
$ ! A sanity check of the file protections shows
$ !
$ DIRECTORY/PROTECTION [.SSH]*.*

Directory DISK$SYS_LOGIN:[MYNAME.SSH]

AUTHORIZED_KEYS.;1   (RWE,RWED,RE,E)
IDENTITY.;1          (RWD,RWD,,)
IDENTITY.PUB;1       (RWE,RWED,RE,E)
KNOWN_HOSTS.;1       (RWD,RWD,,)
RANDOM_SEED.;1       (RWD,RWD,,)

Total of 5 files.
$ !
$ DIRECTORY/PROTECTION SSH.DIR

Directory DISK$SYS_LOGIN:[MYNAME]

SSH.DIR;1            (RWD,RWD,,)

Total of 1 file.
```

Copying SSH2 Key Files

When copying public key files from systems to the system running the SSH server, it is important for the key file to be created in STREAM-LF format or fixed-length 512-byte format on the VMS system. Use DIRECTORY/FULL to determine the format of the key file. The following copy operations should preserve the file format correctly from the specified source systems:

OpenVMS - MultiNet FTP in VMS mode
- DCL COPY
- FTP in BINARY mode
- SCP2

Other O/S - SCP2
- FTP in BINARY mode

If the key file is in VARIABLE format, the server is unable to read the key file successfully, with the result that public-key authentication fails. To convert a VARIABLE format key file to STREAM-LF format, the following FDL file may be used with the RMS CONVERT facility:

FIX_SSH2_KEYS.FDL:

```
TITLE       "File for fixing SSH2 public keys"
IDENT       "OpenVMS FDL Editor"
SYSTEM
            SOURCE                  "OpenVMS"
FILE
            ALLOCATION              64
            BEST_TRY_CONTIGUOUS     yes
            EXTENSION               6
            ORGANIZATION            sequential
RECORD
            BLOCK_SPAN              yes
            CARRIAGE_CONTROL        none
            FORMAT                  stream_LF
             SIZE                        0
```

Port Forwarding

Port forwarding is a mechanism whereby programs that use known TCP/IP ports can have encrypted data forwarded over unsecure connections. This is also known as "tunneling".

If the user is using an authentication agent, the connection to the agent is forwarded automatically to the remote side unless disabled on the command line or in a configuration file. Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either in a configuration file or on the command line using the following qualifier:

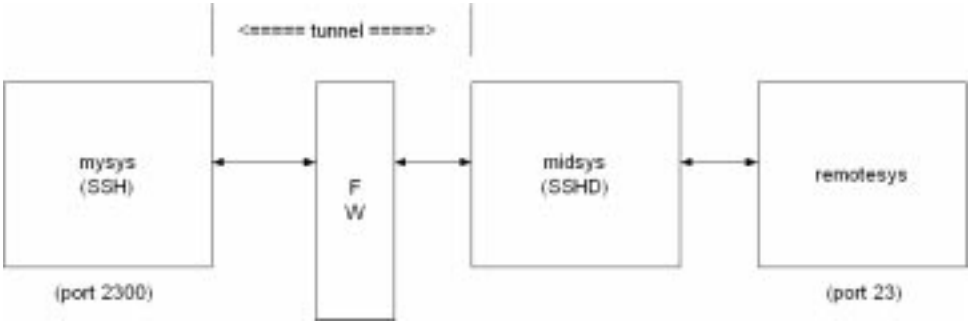
```
/LOCAL_FORWARD=([protocol/]localport:remotehost:remoteport)
```

This causes `localport` on the system the client is running on to be forwarded to `remotehost:remoteport`. The system to which SSH2 connects acts as the intermediary between the two endpoint systems.

The recognized values for the optional protocol are: TCP (default), HTTP(equivalent to TCP), and FTP. The string inside of the ( ) must be quoted " " when the protocol is specified to prevent DCL from interpreting the "/" as the start of a qualifier.

TCP and HTTP don't do anything to the data passing over the connection. When FTP is used, the data stream is examined for FTP, PORT, and PASV commands and their replies so that an SSH2 data stream can be substituted for the FTP data ports.

For example: Use port forwarding to allow a system (`midsys)` to encrypt and forward TELNET sessions between itself (`mysys`) that's outside a corporate firewall to a system (`remotesys`) that is inside a corporate firewall. Note that the use of port 2300 in the examples is arbitrary.



From the DCL prompt on `mysys`:

```
$ SSH midsys /local_forward=(2300:remotesys:23)
```

With the SSH session to `midsys` now active, type in another window on `mysys`:

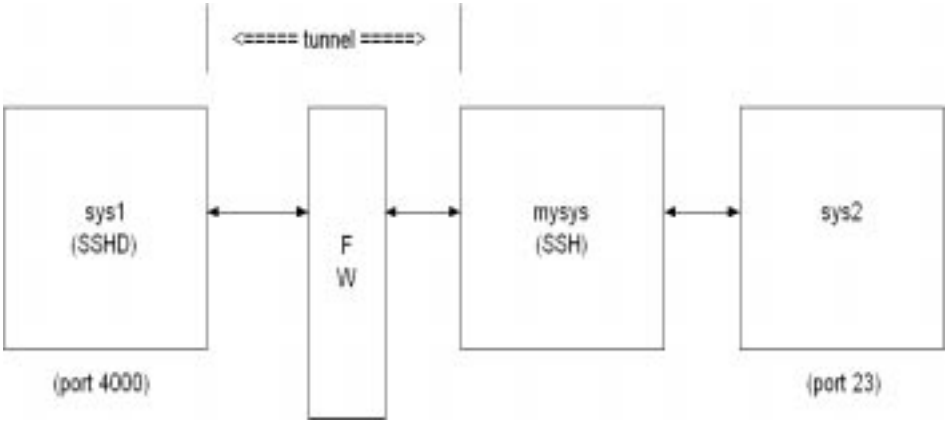
```
$ telnet localhost /port=2300
```

Note! The SSH session must remain active for port forwarding activity.

This causes a connection to `mysys:2300`. The SSH2 client has bound to this port, and will see the connection request. SSH sends an "open channel" request to `midsys`, telling it there's a connect request for port 23 on `remotesys`. `Midsys` will connect to `remotesys:23`, and send back the port information to `mysys`. `Mysys` completes the connection request, and the TELNET session between `mysys` and `remotesys` is now in place, using the tunnel just created through the firewall between `mysys` and `midsys`.

All traffic between `mysys` and `midsys` (through the firewall) is encrypted/decrypted by SSH on `mysys` and SSHD on `midsys`, and hence, is safe. TELNET does not know this, of course, and does not care.

Note that ports can also be forwarded from a localhost to the remotehost that's running SSHD, as illustrated in this figure.



In this example, port 2300 on `mysys` is being forwarded to `remotesys:23`. To do this, use SSH on `mysys`:

```
$ SSH remotesys /local_forward=(2300:remotesys:23)
```

Then, also on `mysys`, type:

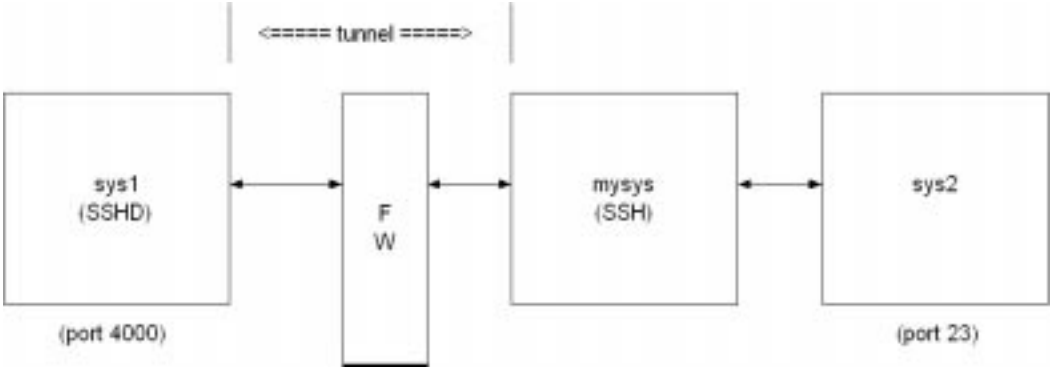
```
$ telnet localhost /port=2300
```

When SSH and SSHD start their dialog, SSHD on `remotesys` connects back to itself, port 23, and the TELNET session is established.

The qualifier

`/REMOTE_FORWARD=([protocol/]remoteport1:remotehost:remoteport2)`

causes `remoteport1` on the system to which SSH connects to be forwarded to `remotehost:remoteport2`. In this case, the system on which the client is running becomes the intermediary between the other two systems.



For example, a user wants to use `mysys` to create a tunnel between `sys1:4000` and `sys2:23`, so that TELNET sessions that originate on `sys1:4000` get tunneled to `sys2` through the firewall. On `mysys,` issue the command:

```
$ SSH sys1 /remote_forward=(4000:sys2:23)
```

Now, on `sys1`, a user could establish a TELNET session to `sys1` by doing:

```
$ TELNET localhost /port=4000
```

The mechanism used for making the TELNET connection (setting up the tunnel) is essentially the same as described in the /LOCAL_FORWARD example above, except that the roles of SSH and SSHD in the dialog are reversed.

Other Files

The files in Table 6-3 are used by SSH. Note that these files generally reside in the [.SSH2] subdirectory from the user's SYS$LOGIN directory. The [.SSH2] subdirectory is created automatically on your local system the first time SSH is executed, and on a remote OpenVMS system the first time an SSH connection is made to that system. File protection for SYS$LOGIN:SSH2.DIR should be (S:RWD, O:RWD, G:, W:).

Note! HP's TCP/IP services do not use the traditional UNIX rhosts and hosts.equiv files; it uses a proprietary format. Therefore, any information added to HP's files via the "ADD PROXY" command must also be manually added to the SSH_DIR:RHOSTS and SSH_DIR:HOSTS.EQUIV files in order for it to be used by SSH for OpenVMS.

Table 6-3 SSH2 Files

File Name	Resides On	Description
[.SSH2]SSH2_CONFIG.	Client System	This is the individual configuration file. This file is used by the SSH2 client. It does not contain sensitive information. The recommended file protection is (S:RWD,O:RWD,G:,W:).
[.SSH2]IDENTIFICATION	Client System	Contains the information about private keys that can be used for public-key authentication, when logging in.
[.SSH2]ID_*alg_bits_seq*	Client System	Contains a private key for authentication. • *alg* is either RSA or DSA • *bits* is the length of the key • *seq* is an incrementing alphabetic value Thus, a key named `ID_DSA_1024_A`. indicates this is a private DSA key 1024 bits long, and it is the first time the key was generated using SSHKEYGEN. A user may have multiple private key files in a directory.

Table 6-3 SSH2 Files (Continued)

File Name	Resides On	Description
[.SSH2]ID_*alg_bits_seq*.PUB	Client System and Server System	Contains a public key for authentication. • *alg* is either RSA or DSA • *bits* is the length of the key • *seq* is an incrementing alphabetic value Thus, a key named ID_DSA_1024_B.PUB indicates this is a public DSA key 1024 bits long, and it is the second time the key was generated using SSHKEYGEN. A user may have multiple public key files in a directory.
[.SSH2.HOSTKEYS]xxx.PUB	Client System	Contains public host keys for all hosts the user has logged into. The files specifications have the format KEY_*port_hostname*.PUB • *port* is the port over which the connection was made • *hostname* is the hostname of the key's host. For example, if tulip.flowers.com was accessed via port 22, the keyfile would be "KEY_22_TULIP_FLOWERS_ COM.PUB". If this file changes on the host (for example, the system manager regenerates the host key), SSH2 will note this and ask if you want the new key saved. This helps prevent man-in-the-middle attacks.

Table 6-3 SSH2 Files (Continued)

File Name	Resides On	Description
[.SSH2]RANDOM_SEED.	Client System	Seeds the random number generator. This file contains sensitive data and MUST have a protection of no more than (S:RWD,O:RWD,G:,W:), and it must be owned by the user. This file is created the first time the program is run and is updated automatically. The user should never need to read or modify this file. On OpenVMS systems, multiple versions of this file will be created; however, all older versions of the file may be safely purged. Use the DCL command: SET FILE /VERSION_LIMIT=n RANDOM_SEED to set a limit on the maximum number of versions of this file that may exist at any given time.
SSH_DIR:.RHOSTS	Server System	Is used in hostbased authentication to list the host/user pairs that are permitted to log in. Each line of the file contains a host name (in the fully-qualified form returned by name servers), and then a user name on that host, separated by a space. This file must be owned by the user, and must not have write permissions for anyone else. The recommended permission is read/write for the user, and not accessible by others.
SSH_DIR:.SHOSTS	Server System	Is used the same way as .RHOSTS.

Table 6-3 SSH2 Files (Continued)

File Name	Resides On	Description
SSH_DIR:HOSTS.EQUIV	Server System	Is used during .rhosts authentication. It contains fully-qualified hosts names, one per line. If the client host is found in this file, login is permitted provided client and server user names are the same. Additionally, successful RSA host authentication is required. This file should only be writable by SYSTEM.
SSH_DIR:SHOSTS.EQUIV	Server System	Is processed exactly as `SSH_DIR:HOSTS.EQUIV`. This file may be useful to permit logins using SSH but not using rshell/rlogin.
SSH2_DIR:SSH2_CONFIG	Client System	This is a system-wide client configuration file. This file provides defaults for those values that are not specified in a user's configuration file, and for users who do not have a configuration file. This file must be world-readable.
MULTINET_SSH2_KNOWNHOSTS_DIR	Server System	Contains public host keys for all hosts the system has logged into. The files specifications have the format `KEY_`*`port_hostname`*`.PUB` • *port* is the port over which the connection was made • *hostname* is the hostname of the key's host. For example, if tulip.flowers.com was accessed via port 22, the keyfile would be "KEY_22_TULIP_FLOWERS_ COM.PUB". If this file changes on the host (for example, the system manager regenerates the host key), SSH will note this and ask if you want the new key saved. This helps prevent man-in-the-middle attacks.

SSHKEYGEN

Generates authentication key pairs. The format of the keys is incompatible between SSH1 and SSH2. Therefore, the correct format keys must be generated for each version of the protocol to be supported.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, you need to generate a new key and copy the corresponding public key to other systems.

Each key may be protected via a passphrase, or it may be left empty. Good passphrases are 10-30 characters long and are not simple sentences or otherwise easily guessable. Note that the passphrase can be changed later, but a lost passphrase cannot be recovered, as a “one-way” encryption algorithm is used to encrypt the passphrase.

Note! The Host Key has no password.

SSH1

```
MULTINET SSHKEYGEN /SSH1 [/BITS=n] [/IDENTITY_FILE=file]
                         [/PASSPHRASE=passphrase] [/COMMENT=comment]
MULTINET SSHKEYGEN /SSH1 /CHANGE_PASSPHRASE [/PASSPHRASE=old_passphrase]
                         [/NEW_PASSPHRASE=new_passphrase]
MULTINET SSHKEYGEN /SSH1 /CHANGE_COMMENT [/PASSPHRASE=passphrase]
                         [/COMMENT=comment]
MULTINET SSHKEYGEN /SSH1 /CHANGE_CIPHER [/IDENTITY_FILE=file]
                         [/PASSPHRASE=passphrase]
MULTINET SSHKEYGEN /SSH1 [/HOST][/BITS=n][/COMMENT=comment]
```

Table 6-4 SSH1 SSHKEYGEN Options

Option	Description
/BITS=*nnn*	Specify key strength in bits (default = 1024).
/CHANGE_PASSPHRASE	Change the passphrase of private key file.
/CHANGE_COMMENT	Change the comment for a key.
/CHANGE_CIPHER	Change the cipher to current default (3DES).
/COMMENT="*comment*"	Provide the comment.
/HOST	Generate the host key.
/IDENTITY_FILE=*file*	Specify the name of the host key file.
/PASSPHRASE=*ppp*	Provide the current passphrase.
/NEW_PASSPHRASE=*ppp*	Provide new passphrase.
/VERSION	Print sshkeygen version number.

SSH2

```
MULTINET SSHKEYGEN /SSH2[/BITS=n][/COMMENT=comment][/KEYTYPE=type]
                   [/KEYS=(key1...keyn)]
                   [/PASSPHRASE=ppp|/NOPASSPHRASE][/STIR=file][/QUIET]
MULTINET SSHKEYGEN /SSH2/HOST
                   [/BITS=n][/COMMENT=comment][/STIR=file][/QUIET]
MULTINET SSHKEYGEN /SSH2/DERIVE_KEY=file
MULTINET SSHKEYGEN /SSH2/EDIT=file
MULTINET SSHKEYGEN /SSH2/FINGERPRINT=file
MULTINET SSHKEYGEN /SSH2/INFO=file [/BASE=n]
MULTINET SSHKEYGEN /SSH2/CONVERT_SSH1=file
MULTINET SSHKEYGEN /SSH2/CONVERT_X509=file
MULTINET SSHKEYGEN /SSH2/CONVERT_PKCS=file
MULTINET SSHKEYGEN /SSH2/EXTRACT_CERTS=file
MULTINET SSHKEYGEN /SSH2/HELP
MULTINET SSHKEYGEN /SSH2/VERSION
```

Table 6-5 SSH2 SSHKEYGEN Options

Option	Description
/BASE=*nnn*	Number base for displaying key info
/BITS=*nnn*	Specify key strength in bits (default = 1024).
/COMMENTS="*comment*"	Provide the comment.
/CONVERT_PKCS=*file*	Convert a PKCS 12 file to an SSH2 format certificate and private key.
/CONVERT_SSH1=*file*	Convert SSH1 identity to SSH2 format.
/CONVERT_X509=*file*	Convert private key from X.509 format to SSH2 format.
/DERIVE_KEY=*file*	Derive the private key given in 'file' to public key.
/EDIT=*file*	Edit the comment/passphrase of the key.
/EXTRACT_CERTS=*file*	Extract certificates from a PKCS 7 file.
/FINGERPRINT=*file*	Dump the fingerprint of file.
/INFO=*file*	Load and display information for 'file'.
/HELP	Print help text.
/HOST	Generate the host key.
/KEYS=(*key1,...,keyn*)	Generate the specified key file(s).
/KEYTYPE=(*dsa* | *rsa*)	Choose the key type: dsa or rsa.
/PASSPHRASE=*ppp*	Provide the current passphrase.
/NOPASSPHRASE	Assume an empty passphrase.
/QUIET	Suppress the progress indicator.
/STIR=file	Stir data from file to random pool.
/VERSION	Print sshkeygen version number.

There is also a comment field in the public key file that is for the 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 `nnn-bit dsa, username@hostname, ddd mm-dd-yyyy hh:mm:ss` when the key is created unless the /COMMENT qualifier is used, and may be changed later using the /EDIT qualifier.

Note! When the /HOST qualifier is used, the /KEYS=(key1,...keyn) qualifier is ignored.

SSHAgent (authentication agent)

```
MULTINET SSHAGENT
```

DESCRIPTION

SSHAGENT is a program that holds authentication private keys. Both SSH1 and SSH2 keys are supported by SSHAGENT. SSHAGENT may be started in the beginning of a login session by including the commands to start it in, for example, LOGIN.COM. It may also be started interactively at any time during a login session.

To start SSHAGENT, one of the three methods may be used:

1. Start it in a separate window:

```
$ MULTINET SSHAGENT
```

2. Spawn it as a subprocess:

```
$ SPAWN/NOWAIT MULTINET SSHAGENT
```

3. Run it in a detached process:

```
$ RUN/DETACHED/OUTPUT=AGENT.OUT/PROCESS_NAME="SSH_AGENT"/INPUT=NLA0:
  SSH_EXE:SSH-AGENT2
```

The agent is used for Publickey Authentication when logging to other systems using SSH. A connection to the agent is available to all programs run by all instances of the user on a specific system. The name of the mailbox used for communicating with the agent is stored in the MULTINET_SSH_AGENT_ *username* logical name. Note that while the agent mailbox is accessible only by the user that starts the agent, a user with sufficient VMS privileges could access the agent mailbox and steal or modify keys currently loaded into the agent (although, the keys as stored on disk cannot be modified simply by accessing the agent).

The agent does not have any private keys initially. Keys are added using SSHADD. When executed without arguments, SSHADD adds the user’s identity files. If the identity has a passphrase, SSHADD asks for the passphrase. It then sends the identity to the agent. Several identities can be stored in the agent; the agent can use any of these identities automatically.
`$ MULTINET SSHADD /LIST` displays the identities currently held by the agent. The idea is that the agent is run on the user's workstation.

FILES

[.SSH]IDENTITY in SYS$LOGIN:	Contains the 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 is used to encrypt the private part of this file. This file is not used by SSHAGENT, but is added to the agent using SSHADD at login.

SSHADD

Adds identities for the authentication agent.

```
MULTINET SSHADD [OPTIONS] [FILE[,FILE,FILE]]
```

DESCRIPTION

SSHADD adds identities to SSHAGENT, the authentication agent. When run without arguments, SSHADD adds the file [.SSH]IDENTITY. Alternative file names can be given on the command line. If any file requires a passphrase, SSHADD asks for the passphrase from the user.

The authentication agent must be running and must have been executed by the user for SSHADD to work.

“File” is an identity or certificate file. If no file is specified, the files in the users[.SSH2] directory are used.

OPTIONS

/HELP	Display help text.
/LIST	List all identities currently represented by the agent.
/LOCK	Lock the agent with a password.
/NOSSH1	Agent cannot use SSH1 keys.
/PURGE	Remove all identities from the agent.
/REMOVE	Remove the identity from the agent.
/TIMEOUT=*n*	Agent should delete this key after the timeout value (in seconds) expires.
/UNLOCK	Unlock the locked agent.
/URL	Give key to the agent as a URL.

FILES

These files exist in SYS$LOGIN:

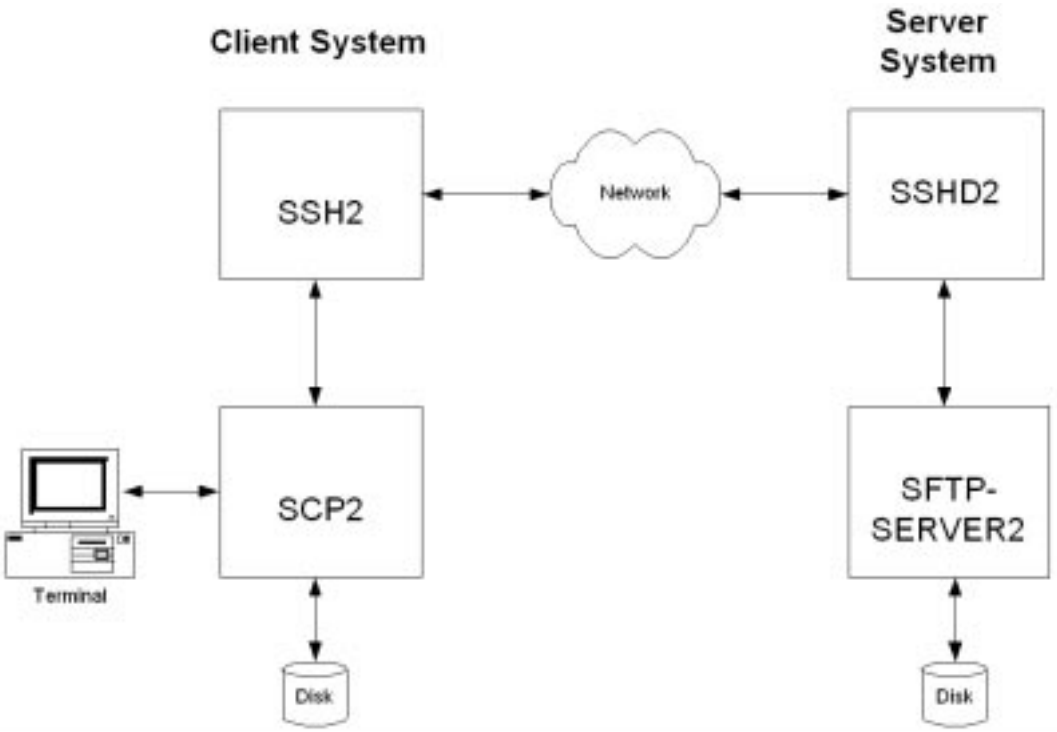
[.SSH]IDENTITY	Contains the RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key. That passphrase is used to encrypt the private part of this file. This is the default file added by SSHADD when no other files have been specified. If SSHADD needs a passphrase, it reads the passphrase from the current terminal if it was run from a terminal. If SSHADD does not have a terminal associated with it but DECW$DISPLAY is set, it opens an X11 window to read the passphrase.
[.SSH]IDENTITY.PUB	Contains the public key for authentication. The contents of this file should be added to [.SSH]AUTHORIZED_KEYS on all systems where you want to log in using RSA authentication. There is no need to keep the contents of this file secret.
[.SSH]RANDOM_SEED	Seeds the random number generator. This file should not be readable by anyone but the user. This file is created the first time the program is run, and is updated every time SSHKEYGEN is run.

Chapter 7

Secure File Transfer

The Secure File Transfer mechanism consists of three programs — the client program SCP2, which includes an embedded SFTP server for local file access, and SFTP-SERVER2 and SCP-SERVER1, which run on the remote system to access the file. SCP2 communicates with SSH2 for authentication and data transport (which includes encryption) to remote systems, SFTP-SERVER2 communicates with SSHD2 for data transport.

The following diagram illustrates the relationship among the client and server portions of an SCP2 file transfer:



SCP file transfers are different from FTP file transfers. With FTP a file can be transferred as ASCII, BINARY, RECORD, or in VMS Plus mode (if Compaq TCP/IP Services for OpenVMS is in use). In SCP there is one specified format — BINARY. Also, the defined syntax for a file specification is UNIX syntax. Due to these restrictions, files that are transferred from dissimilar systems may or may not be useful. Process Software has used methods available in the protocol to attempt to improve the chances that files will be useful upon transfer. The SSH File Transfer Protocol is an evolving specification, and some implementations may not support all options available in the protocol, or worse, not tolerate some optional parts of later versions of the protocol.

Process Software has used the defined extensions in the protocol to transfer information about the VMS file header characteristics such that when a file is transferred between two VMS systems running MultiNet v4.4, TCPware v5.6, and/or SSH for OpenVMS, the file header information will also be transferred and the file will have the same format on the destination system as it had on the source system. Also, when a file is transferred to a non-VMS system, a method has been provided to translate those files that can be translated into a format that will be usable on the remote system. Files that are transferred from non-VMS systems are stored as stream files on the VMS system, which provides compatibility for text files from those systems.

SCP-SERVER1

The SCP-SERVER1 program is used when a system with OpenSSH initiates an SCP command. OpenSSH uses RCP over SSH2 instead of the SFTP protocol. SCP-SERVER1 will always translate VMS text files (if possible) when copying a file from VMS. Translated VMS text files may have some trailing nulls at the end of them, due to the RCP protocol not being able to tolerate a file that comes up short of the reported size. SCP-SERVER1 (and SFTP-SERVER2) use sophisticated methods to estimate the amount of user data in the file to minimize this. On ODS-5 disks the estimation routine uses the file size hint if it is valid. On ODS-2 disks (and ODS-5 without a valid size hint), the size of the file and file characteristics are used to estimate the amount of user data. The method provides as accurate an estimate as possible without actually reading the file and never underestimates the amount of data in the file. Underestimating would cause significant problems as the programs use the size of the file to determine how much data to expect.

SCP2

Usage

```
SCP2 [qualifiers] [[user@]host[#port]::]file [[user@]host[#port]::]file
```

Note! The source and destination file specification must be quoted if they contain a user specification or a non-VMS file specification.

Qualifiers

Table 7 -1 SCP Qualifiers

Qualifier	Description
/BATCH	Starts SSH2 in batch mode. Authentication must be possible without user interaction.
/BUFFER_SIZE=*integer*	Number of bytes of data to transfer in a buffer. Default is 7500. Minimum value is 512.
/CIPHER=*(cipher-1,...,cipher-n)*	Selects an encryption algorithm(s).
/COMPRESS	Enables SSH data compression.
/CONCURRENT_REQUEST=*integer*	Number of concurrent read requests to post to the source file. Default is 4.
/DEBUG=*level*	Sets a debug level. (0-99)
/DIRECTORY	Forces the target to be a directory.
/HELP	Displays the help text.
/IDENTITY_FILE=*file*	Identifies the file for public key authentication.
/PORT=*number*	Tells SCP2 which port SSHD2 listens to on the remote machine.
/PRESERVE	Preserves file attributes and timestamps.
/NOPROGRESS	Does not show progress indicator.
/QUIET	Does not display any warning messages.
/RECURSIVE	Processes the entire directory tree.
/REMOVE	Removes the source files after copying.
/TRANSLATE_VMS= *(ALL, NONE, VARIABLE, FIXED, VFC)*	Selects the VMS text files to be translated (default=ALL).
/VERBOSE	Displays verbose debugging messages. Equal to "/debug=2".

Table 7 -1 SCP Qualifiers (Continued)

Qualifier	Description
/VERSION	Displays the version number only.
/VMS	Negotiates the ability to transfer VMS file information.

Note! /VMS and /TRANSLATE_VMS are mutually exclusive

File Specifications

The source and destination strings are changed to lowercase unless they are enclosed in quotes, in which case they are left the same. File specification must be in UNIX format for remote systems, unless the remote system is running TCPware 5.6, MultiNet v4.4, or SSH for OpenVMS, and /VMS or /TRANSLATE_VMS (source files only) are used. UNIX format file specifications need to be enclosed in quotes (") if they contain the / character to prevent the DCL parsing routines from interpreting the string as a qualifier.

Qualifiers

/BATCH

Starts SSH2 in BATCH mode. When SSH2 is running in BATCH mode it does not prompt for a password, so user authorization is accomplished by Public-Key authentication.

/BUFFER_SIZE=integer

Number of bytes of data to transfer in a buffer. Default is 7500.

/CIPHER=(cipher,...,cipher-n)

Lets you select which SSH2 cipher to use.

/COMPRESS

Enables SSH2 data compression. This can be beneficial for large file transfers over slow links. The compression level is set by the client configuration file for SSH2.

/CONCURRENT_REQUEST=integer

Number of concurrent read requests to post to the source file. Default is 4.

/DEBUG

Enables debugging messages for SCP2 and SSH2. Higher numbers get more messages. The legal values are between 0 (none) and 99. Debugging for SFTP-SERVER2 is enabled via the MULTINET_SSH_SFTP_SERVER_DEBUG logical.

/DIRECTORY

Informs SCP2 that the target specification should be a directory that the source file(s) will be put in. This qualifier is necessary when using wildcards in the source file specification, or /RECURSIVE.

/HELP

Displays command qualifier list and parameter format.

/IDENTITY_FILE=file

Specifies the identity file that SSH2 should use for Public-Key authentication.

/PORT=number

Specifies the port that SSH2 uses on the remote system. Note that if both the source and destination files are remote, this value is applied to both. If SSH2 is available on different ports on the two systems, then the #port method must be used.

/PRESERVE

Sets the Protection, Owner (UIC), and Modification dates on the target file to match that of the source file. /PRESERVE is not very useful when the target machine is a VMS system as VMS does not provide runtime library calls for setting the file attributes (owner, protection) and timestamps. Note that the VMS modification date (not the creation date) is propagated to the remote system. When files are copied between two VMS systems and /VMS is used /PRESERVE is implied and the process of transferring VMS attributes preserves the information about the protection, dates, and file characteristics. The file access time is not adjusted for the difference between local time and GMT.

/NOPROGRESS

SCP2, by default, updates a progress line at regular intervals when it is run interactively to show how much of the file has been transferred. This qualifier disables the progress line.

/QUIET

Disables warning messages. Note that it does not disable warning messages from SFTP-SERVER2, which return on the error channel.

/RECURSIVE

Copies all of the files in the specified directory tree. Note that the top level directory on the local system is not created on the remote system. When /VMS is used, all versions of the files are copied.

When /VMS is not used, only the most recent version is copied.

/REMOVE

Deletes the source files after they have been copied to the remote system.

/TRANSLATE_VMS

Translates VMS text files in the copying process to byte streams separated by linefeeds because the defined data transfer format for SCP2 is a binary stream of bytes.

/TRANSLATE_VMS is only applicable to the source specification. If a remote source file is specified, then that system must be running MultiNet v4.4, TCPware 5.6, or SSH for OpenVMS. If /TRANSLATE_VMS is specified with no value, then VARIABLE, FIXED, and VFC (Variable, Fixed Control) files are translated to stream linefeed files. If the value is NONE, no files are translated. VARIABLE, FIXED, and VFC can be combined in any manner. The SFTP-SERVER2 process also uses the value of the logical MULTINET_SFTP_TRANSLATE_VMS_FILE_TYPES to determine which files should be translated. This is a bit mask with bit 0 (1) = FIXED, bit 1 (2) = VARIABLE, and bit 2 (4) = VFC. These values can be combined into a number between 0 and 7 to control which files are translated.

Note! Due to the structure of the programs, the SCP2 program uses this logical if the /TRANSLATE_VMS qualifier has not been specified.

/VERBOSE

Displays debugging messages that allow the user to see what command was used to start up SSH and other basic debugging information. Note that debugging information can interfere with the normal display of the progress line. Equivalent to /DEBUG=2.

/VERSION

Displays the version of the base SCP2 code.

/VMS

Transfers VMS file information similar to that transferred in OVMS mode in FTP such that VMS file structure can be preserved. All of the information transferred in FTP OVMS mode is transferred along with the file creation date and protection. If the file is a contiguous file, and it is not possible to create the file contiguously, and the logical MULTINET_SFTP_FALLBACK_TO_CBT has the value of TRUE, YES, or 1, SFTP-SERVER2 attempts to create the file Contiguous, Best Try. VMS mode is only available with SCP2 provided in MultiNet v4.4 and TCPware 5.6, and SSH for OpenVMS. /VMS also modifies the usual sequence of operations that SCP2 does such that a new version of the file is created if there are existing versions. Without /VMS, the most recent version of the target file is deleted (if it exists) before the new file is written. This is to accommodate systems that do not handle multiple versions of files.

The logical name MULTINET_SCP2_VMS_MODE_BY_DEFAULT can be defined to TRUE, YES, or 1 to specify that /VMS should be the default unless /NOVMS or /TRANSLATE_VMS are specified. /VMS and /TRANSLATE_VMS can not be used on the same command line. If /VMS is not specified, but the logical is set to enable it by default, a /TRANSLATE_VMS on the command line will take precedence.

Note that even though SCP2 & SFTP-SERVER2 pass the request for VMS file transfers or to translate a VMS file in a manner that is consistent with the protocol specification, other implementations may not handle this information well. Since there is no error response present at that point in the protocol, the program hangs. To prevent it from hanging forever, the logical MULTINET_SCP2_CONNECT_TIMEOUT is checked to see how long SCP2 should wait for a response when establishing the connection. The format for this logical is a VMS delta time. The default value is 2 minutes. If SCP2 times out before a connection is established with SFTP-SERVER2 and /VMS or /TRANSLATE_VMS were specified, a warning message is displayed and the initialization is tried again without the request for VMS information (or /TRANSLATE_VMS). This retry is also subject to the timeout, and if the timeout happens again, then SCP2 exits. This helps for implementations that ignore the initialization message when information they do not recognize is present; implementations that abort will cause SCP2 to exit immediately.

Logicals

The following logicals apply to both SCP2 and SFTP-SERVER:

MULTINET_SFTP_FALLBACK_TO_CBT
MULTINET_SFTP_TRANSLATE_VMS_FILE_TYPES
MULTINET_SFTP_RETURN_ALQ

MULTINET_SFTP_FALLBACK_TO_CBT

When defined to TRUE, YES, or 1 and a VMS file transfer is being performed, this logical creates a Contiguous file if that file has Contiguous characteristics. The file will be created as Contiguous Best Try if there is insufficient space to create it as Contiguous.

MULTINET_SFTP_TRANSLATE_VMS_FILE_TYPES

This is a bit mask that determines which VMS file types should be translated when not operating in VMS mode.

- Bit 0 (1) = FIXED
- Bit 1 (2) = VARIABLE
- Bit 2 (4) = VFC

The values are:

- 0 (zero) = NONE
- 7 = ALL

Note that this logical affects SCP2 as well as the server, as SCP2 has the server built into it for handling local file access.

MULTINET_SCP2_CONNECT_TIMEOUT

This logical defines a number specifying how long SCP2 should wait for a response to the INITIALIZE command from the server program. This is a VMS delta time number. The default is 2 minutes.

MULTINET_SCP2_VMS_MODE_BY_DEFAULT

When defined to TRUE, YES, or 1, this logical chooses the /VMS qualifier if /TRANSLATE_VMS or /NOVMS has not been specified.

MULTINET_SFTP_RETURN_ALQ

When defined to TRUE, YES, or 1 and files are being transferred in VMS mode, this logical includes the Allocation Quantity for the file in the file header information. This is disabled by default because copying a small file from a disk with a large cluster size to a disk with a small cluster size causes the file to be allocated with more space than necessary. You have the option of retaining the allocated size of a file if it was allocated the space for a reason. Some combinations of file characteristics require that the Allocation Quantity be included in the file attributes; this is handled by SCP2/SFTP-SERVER2.

MULTINET_SSH_SCP_SERVER_DEBUG

Enables debugging messages for the SCP-SERVER1 image that provides service to SCP commands that use the RCP over SSH2 protocol (OpenSSH). When this is defined, the file SCP-SERVER.LOG is created in the user's login directory. These files are not purged. Larger values yield more debugging information.

MULTINET_SSH_SFTP_SERVER_DEBUG

Enables debugging messages for the SFTP-SERVER2 image that provides service to SCP2 commands that use the SFTP protocol. When this is defined, the file SFTP-SERVER.LOG is created in the user's login directory. These files are not purged. Larger values yield more debugging information.

FTP over SSH

SSH2 can be used to set up port forwarding that can be used for FTP. This allows users to use the richness of the FTP command set to access files on a remote system and have their control and data information encrypted. The command format to set up the SSH port forwarding is:

```
$ ssh <remote_host_name>/local_forward=
(“““ftp/<forwarded_port_number>:localhost:21”””)
```

The usual SSH authentication mechanisms come into play, so there may be a request for a password and a terminal session is established to the remote host. As long as this terminal session is alive, other users on the local system can use FTP to access the remote system over an encrypted channel. The location of the quotes is important, as it is necessary to prevent DCL from interpreting the / in

the local forwarding information as the start of a new qualifier, and SSH2 does not know or expect to find the ( ) around the forwarding information. Note that the “localhost” inside of the forwarding string is important, as it will make the connection to FTP on the remote system come from localhost, which will then allow FTP to open the data port.

When a user desires to use an encrypted FTP connection, the following OPEN command would be issued to the TCP/IP Services for OpenVMS FTP client:

```
OPEN LOCALHOST <forward_port_number>
```

Normal FTP authentication takes place and multiple FTP sessions may use a single forwarded port. The FTP protocol filter in SSH2 scans the FTP command stream for the FTP PORT and PASV commands and their replies, and makes substitutions in these commands and replies to use a secure data stream through the SSH2 session that has been set up.

To allow a single system to act as a gateway between two networks, add /ALLOW_REMOTE_CONNECT to the SSH command that initiates the connection. This command will establish an encrypted FTP session with the remote host that the SSH connection is sent to.

Chapter 8

Monitoring and Controlling SSH

SSH for OpenVMS provides utilities for monitoring and controlling the SSH server environment. The following topics describe the utilities, their capabilities, and their use.

Controlling SSH Server Functions

The following control functions are available for the SSH servers:

- Startup
- Shutdown
- Restart
- Set debug level

The SSHCTRL Utility

The SSHCTRL utility is used to perform all but the startup function. For the startup function, the SYS$STARTUP:PSCSSH$STARTUP.COM file is used.

Usage: SSHCTRL <*operation*> [*options*]

Table 8-1 shows the various operations that can be used with the SSHCTRL utility.

Table 8-1 SSHCTRL Utility Operations

Operation	Description
SET /DEBUG=*n*	Set debug level (0 = no debug)
SHOW	Show session information.
SHOW /ALL	Show all sessions. This is the default if no switch is used with the SHOW keyword.
SHOW /USER=*username*	Show sessions for <*username*>.
SHOW /HOST=*address*	Show sessions for <*address*>.
SHUTDOWN	Stop all SSH server sessions.
RESTART	Stop/restart SSH server.
HELP	Display help text.
VERSION	Display version information.

Starting the SSHD Master Process

```
$ @SYS$STARTUP:PSCSSH$STARTUP
Starting SSH for OpenVMS...
%RUN-S-PROC_ID, identification of created process is 22C000AD
$
```

Shutting down the SSHD Master Process

This function is used to stop the SSHD Master process on the system, so it won't accept new connections. Note that shutting down the SSHD Master process will also terminate all outstanding SSH server sessions on the system. OPER privilege is required to shut down the SSHD Master process and its servers.

```
$ SSHCTRL SHUTDOWN
Shutting down SSH for OpenVMS...
$
```

Restarting the SSHD Master Process

Restarting the SSHD Master process is required after the CNFSSH utility is used to modify the existing configuration. Note that restarting the SSHD Master process will terminate all outstanding SSH server sessions on the system. OPER privilege is required to restart the SSHD Master process.

```
$ SSHCTRL RESTART
Shutting down SSH for OpenVMS...
Starting SSH for OpenVMS...
%RUN-S-PROC_ID, identification of created process is 22C000B8
$
```

Changing the Server Debug Level

The server debug level is changed using SSHCTRL. The debug level controls the amount of debug information written to the SSH_LOG:SSHD.LOG file for each server instance. This may be a value from 0 (no debug) to 50 (maximum debug). Process Software recommends this value not be set above 5 without instructions from Process Software, as the amount of debug information written to the log at higher levels can severely impact both the SSH server performance and the server host disk resources.

Note that setting the debug level only affects new server processes which are started after setting the level. Currently active servers use the debug level set when they were started. OPER privilege is required to change the debug level.

```
$ SSHCTRL SET/DEBUG=4
SSHCTRL-S-DEBUGSET - old debug level = 2, new debug level = 4
$
```

Displaying SSH Server Utilization

The SSHCTRL SHOW command is used to display the active SSH server sessions on a system. It can display all users (/ALL), users with a specific username (/USER=dogbert), or users with sessions that originate from a specific host (/HOST=192.168.29.248).

Normally, a user may only display the sessions with the same UIC as his own. GROUP privilege is required to display the sessions with UICs in the same group as the user. WORLD privilege is required to display all other servers.

For each session, the display is of the following form:

```
Process “<processname>” (pid<pid>) - an <ssh1|ssh2>session
   User = <login username>
   From system <originating address>port<originating port>
   Started: <date/time session was started>
   Bytes in: <count> out: <count> (from child process <count>)
Child process = “<process name>”(pid<pid>) - an <type> session
```

```
    PTD Device = <_FTAnn:>
    Started <date/time this child started>
```

Note that SSH2 provides the capability for one server to handle multiple child sessions. The child sessions may be a mixture of interactive SSH2 sessions and file transfer (SCP/SFTP) sessions. Currently, only the F-Secure SSH Client for Windows has this capability.

In Example 8-1, a display of all users on the system is done. Note that server “SSHD 0003” actually has six active child processes.

Example 8-1 Showing All Active Server Sessions

```
$ SSHCTRL SHOW /ALL

SSHD Master PID = 22C000B8

Debug level is set to 4

Process "SSHD 0000" (pid 22C000B9) - an SSH2 session
      User = dilbert
      From system 192.168.29.52 port 49152
      Started: 05/03/2002 03:05:22
      Bytes in: 262  out: 0  (from child process: 15100)

      Child process = "DILBERT_@FTA4" (pid 22C000BA) - an SSH2 session
         PTD Device = _FTA4:
         Started: 05/03/2002 03:05:35

Process "SSHD 0003" (pid 22C000BF) - an SSH2 session
      User = DOGBERT
      From system 192.168.29.50 port 1129
      Started: 05/03/2002 03:07:46
      Bytes in: 0  out: 0  (from child process: 55215)

      Child process = "DOGBERT_@FTA9" (pid 22C000C0) - an SSH2 session
          PTD Device= _FTA9:
          Started: 05/03/2002 03:07:54
      Child process = "SSHD 0003A SFTP" (pid 22C000C1) - an SFTP-SERVER2 session
          PTD Device = _FTA10:
          Started: 05/03/2002 03:07:55
      Child process = "DOGBERT_@FTA11" (pid 22C000C2) - an SSH2 session
          PTD Device = _FTA11:
          Started: 05/03/2002 03:07:57
      Child process = "SSHD 0003B SFTP" (pid 22C000C3) - an SFTP-SERVER2 session
          PTD Device = _FTA12:
          Started: 05/03/2002 03:08:00
      Child process = "SSHD 0003C SFTP" (pid 22C000C4) - an SFTP-SERVER2 session
          Device = _FTA13:
          Started: 05/03/2002 03:08:07
      Child process = "DOGBERT_@FTA14" (pid 22C000C5) - an SSH2 session
          PTD Device = _FTA14:
          Started: 05/03/2002 03:08:09

Process "SSHD 0004" (pid 22C000C6) - an SSH1 session
      User = CATBERT
      From system 192.168.29.51 port 1023
```

```
      Started: 05/03/2002 03:08:29
      Bytes in: 0  out: 537  (from child process: 17)


      Child process = "CATBERT_@FTA15" (pid 22C000C7) - an SSH1 session
          PTD Device = _FTA15:
          Started: 05/03/2002 03:08:29
```

Example 8-2 illustrates showing the sessions that originate from a specific TCP/IP address:

Example 8-2 Showing Sessions From a Specific Address

```
$ SSHCTRL SHOW /HOST=192.168.29.51

SSHD Master PID = 22C000B8

Debug level is set to 4

Process "SSHD 0004" (pid 22C000C6) - an SSH1 session
      User = CATBERT
      From system 192.168.29.51 port 1023
      Started: 05/03/2002 03:08:29
      Bytes in: 0  out: 537  (from child process: 17)

      Child process = "CATBERT_@FTA15" (pid 22C000C7) - an SSH1 session
        PTD Device = _FTA15:
        Started: 05/03/2002 03:08:29
```

A

B

C

E

H

I

K

L

M

N

P

R

S

T

U

X

Reader's Comments
SSH for OpenVMS Version 1.0 Administration and User's Guide

Your comments and suggestions will help us to improve the quality of our future documentation. Please note that this form is for comments on documentation only.

I rate this guide's:	**Excellent**	**Good**	**Fair**	**Poor**
Accuracy	o	o	o	o
Completeness (enough information)	o	o	o	o
Clarity (easy to understand)	o	o	o	o
Organization (structure of subject matter)	o	o	o	o
Figures (useful)	o	o	o	o
Index (ability to find topic)	o	o	o	o
Ease of use	o	o	o	o

1. I would like to see more/less:_____

2. Does this guide provide the information you need to perform daily tasks?_____

3. What I like best about this guide:_____

4. What I like least about this guide:_____

My additional comments or suggestions for improving this guide:_____
_____

I found the following errors in this guide:

Page	**Description**
_____	_____
_____	_____

Please indicate the type of user/reader that you most nearly represent:

System Manager	o	Educator/Trainer	o
Experienced Programmer	o	Sales	o
Novice Programmer	o	Scientist/Engineer	o
Computer Operator	o	Software Support	o
Administrative Support	o	Other (please specify)	o _____

Name:_____ Dept._____
Company:_____ Date_____
Mailing Address:_____

After filling out this form, FAX or mail it to:
Process Software, 959 Concord Street, Framingham, MA 01701-4682
Attention: Technical Publications Group FAX 508-879-0042 e-mail:techpubs@process.com