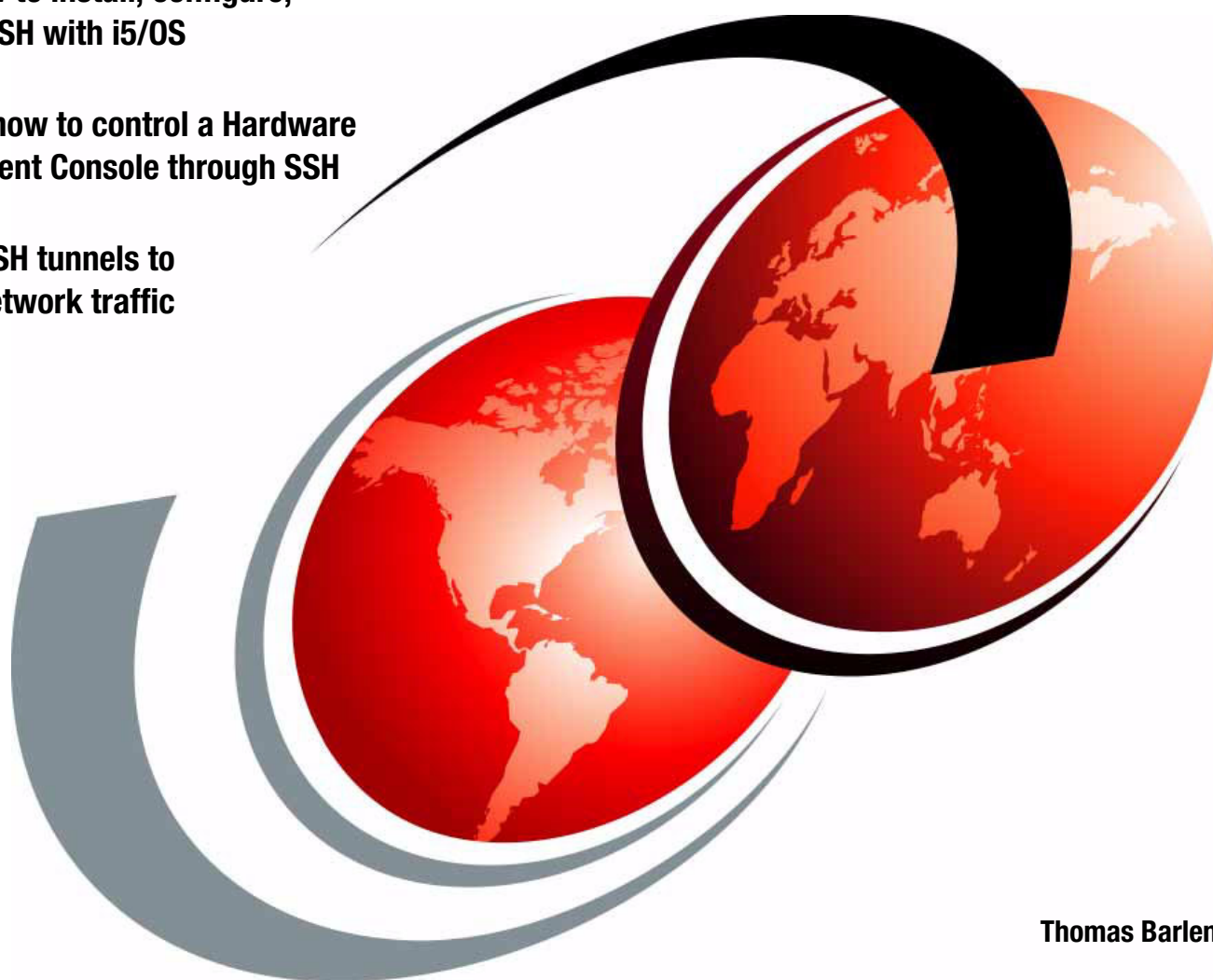


# Securing Communications with OpenSSH on IBM i5/OS

Learn how to install, configure, and use SSH with i5/OS

Discover how to control a Hardware Management Console through SSH

Explore SSH tunnels to protect network traffic



Thomas Barlen





International Technical Support Organization

**Securing Communications with OpenSSH on IBM i5/OS**

July 2006

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (July 2006)**

This edition applies to Version 5, Release 4, Modification 0 of IBM i5/OS (product number 5722-SS1).

**© Copyright International Business Machines Corporation 2006. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	v
Trademarks .....	vi
<b>Preface</b> .....	vii
The person who wrote this Redpaper .....	vii
Become a published author .....	viii
Comments welcome .....	viii
<b>Chapter 1. Introduction to OpenSSH for i5/OS</b> .....	1
1.1 OpenSSH tools and files .....	2
1.2 i5/OS implementation .....	4
1.3 Installing the IBM Portable Utilities for i5/OS license program .....	5
1.4 Redpaper example environment .....	5
1.5 Additional information .....	6
<b>Chapter 2. Setting up and running the sshd daemon</b> .....	7
2.1 Setting up the sshd daemon .....	8
2.1.1 Modifying the sshd daemon system configuration .....	8
2.2 Starting the sshd daemon with Submit Job (SBMJOB) .....	9
2.3 Starting the sshd daemon in a dedicated subsystem environment .....	9
<b>Chapter 3. Establishing an SSH session</b> .....	13
3.1 Preparing the user environment .....	14
3.1.1 Creating the home directory .....	14
3.1.2 Setting the home directory permissions .....	14
3.2 Using SSH between i5/OS environments .....	15
3.2.1 Using the ssh utility to run commands remotely .....	17
3.3 Using SSH from other platforms to i5/OS .....	18
3.3.1 Using PuTTY to establish an SSH connection to i5/OS .....	18
<b>Chapter 4. File transfer and public key authentication with OpenSSH</b> .....	21
4.1 Setting up public key authentication .....	22
4.2 Using public key authentication with scp to transfer files .....	26
4.2.1 Running the scp command in batch mode .....	27
4.3 Exploiting public key authentication with ssh .....	31
<b>Chapter 5. Protecting traffic with SSH tunnels</b> .....	33
5.1 Setting up an SSH tunnel between i5/OS environments .....	34
5.2 Setting up an SSH tunnel between a workstation and i5/OS .....	36
5.3 Automating the tunnel session start .....	45
<b>Chapter 6. Using SSH to control your HMC</b> .....	47
6.1 Setting up SSH on the HMC .....	48
6.2 Setting up public key authentication .....	50
6.3 Moving resources between partitions using SSH in i5/OS .....	52
<b>Abbreviations and acronyms</b> .....	57
<b>Related publications</b> .....	59
IBM Redbooks .....	59

Online resources . . . . .	59
How to get IBM Redbooks . . . . .	60
Help from IBM . . . . .	60
<b>Index . . . . .</b>	<b>61</b>

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

*The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law.* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:


This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AFS®  
AIX®  
AS/400®  
eServer™  
i5/OS®

IBM®  
iSeries™  
OfficeVision/400™  
OS/400®  
POWER5™

Redbooks™  
Redbooks (logo) ™  
System i™  
System i5™  
System p5™

The following terms are trademarks of other companies:

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.



# Preface

The OpenSSH open source product is widely used to securely access command line shells remotely. Secure Shell (SSH) tunneling or port forwarding capabilities allow users to establish a secure link for data traffic that otherwise flows in the clear over communication links. Additional utilities, such as **scp** and **sftp**, provide secure file transfer services. On the IBM® System i™ platform, OpenSSH is shipped as part of the license program option IBM Portable Utilities for i5/OS® and is available for systems that run V5R3 and later.

This IBM Redpaper introduces you to the OpenSSH implementation and the included utilities in IBM i5/OS. It teaches you how to use SSH in i5/OS as a server (daemon) and as a client. It explains how you can set up public key authentication for better protection during authentication.

In addition, this paper discusses the configuration for port forwarding from i5/OS as well as from Microsoft® Windows®. It also explains how you can use SSH to control a dynamic logical partitioning (DLPAR) environment. Plus it explores how SSH can be used from i5/OS to control Hardware Management Console (HMC) tasks, such as dynamic resource allocation.

The purpose of this Redpaper is to show you how to set up and use OpenSSH in an i5/OS environment, not to cover all capabilities provided with OpenSSH. For a complete list of functions and online help, visit the OpenSSH Web site at the following address:

<http://www.openssh.org>

## The person who wrote this Redpaper

This Redpaper was produced by the following specialist, who was working at the International Technical Support Organization (ITSO), Rochester Center.



**Thomas Barlen** is an IBM Certified Consulting IT Specialist in IBM Germany for the System i platform in IBM Systems & Technology Group. From 1999 until the end of 2002, Thomas was assigned to the IBM ITSO center in Rochester, MN. He writes extensively, teaches IBM classes, and is a frequent speaker at conferences worldwide on all areas of System i communications, On Demand Business, single signon, and security. Prior to his start in the ITSO in 1999, he worked in AS/400® software support and as a systems engineer in IBM Germany. He has over 17 years of experience in AS/400 networking, security, and systems management.

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks™ in one of the following ways:

- Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. JLU Building 107-2  
3605 Highway 52N  
Rochester, Minnesota 55901-7829



# Introduction to OpenSSH for i5/OS

The Secure Shell (SSH) was originally designed to provide mainly a secure remote login to and a file transfer utility between remote computers. The communication protocol is an SSH protocol that runs on top of Transmission Control Protocol (TCP). Currently, two protocols are supported, SSH1 and SSH2. These two protocols are entirely different and therefore are not compatible. Today, a company named SSH Communications Security, the original developer of SSH, maintains these protocols. The SSH transport layer protocol is described in draft form on the Web at the following address:

<http://www.ietf.org/html.charters/secsh-charter.html>

In this chapter, we explain the tools and files that are provided with OpenSSH as well as the implementation of OpenSSH on i5/OS and installation of the IBM Portable Utilities for i5/OS licensed program. In addition, we present an overview of the environment used in this Redpaper.

## 1.1 OpenSSH tools and files

OpenSSH is the Open Source version of SSH and provides a set of tools that allows secure communications between two communication partners using the SSH protocol. The following tools and files are provided with OpenSSH:

- ▶ **ssh** client utility (basic **rlogin** or **rsh**-type client program)

The **ssh** client utility is a program for logging into a remote system and for performing commands on a remote system. It is intended to replace the **rsh** and **rlogin** utilities, which do not provide a secure connection. It also provides secure encrypted communications between two systems over an untrusted network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.

In the UNIX® and Linux® world, **ssh** is a common utility to securely communicate to a remote system. It is also used as a client to establish an SSH connection to the Hardware Management Console (HMC) with the IBM POWER5™ technology-based IBM System i5™ and System p5™ platforms. The SSH connection with the HMC provides administrators with a command line interface.

**ssh** connects and logs into the specified host name (with optional user name). The user must prove their identity to the remote system using one of several authentication methods depending on the protocol version that is used.

A popular graphical **ssh** client is the PuTTY client. You can download it freely from the Internet at the following Web address:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

This Web address takes you to the same page of the PuTTY programmer. This **ssh** client is published under the MIT license, which is compatible with the GNU GPL license. It allows both individuals and companies to use the client without restriction.

- ▶ **sshd** daemon (permits you to login)

The **ssh** daemon (**sshd**) is the daemon (server) program for **ssh**. Together these programs replace **rlogin** and **rsh** and provide a secure encrypted communications link between two untrusted hosts over an insecure connection.

The **sshd** daemon listens by default on port 22 for connections from clients. It is normally started at IPL time. It spawns a new daemon process for each incoming connection. This implementation of the **sshd** daemon supports both SSH protocol Versions 1 and 2 simultaneously.

- ▶ **ssh\_config** client configuration file

There is a system wide configuration file for **ssh** client settings. These settings are used for every **ssh**, **sftp**, and **scp** client request. The system-wide configuration file, with the name **ssh\_config**, is stored in the /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc directory.

- ▶ **sshd\_config** daemon configuration file

Customizing the **sshd** configuration file is optional. It is required only when the default values meet your requirements. The configuration file is in the integrated file system directory /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc. The configuration file name is **sshd\_config**. By default, it has all configuration directives as comments listed. The values shown represent the default settings.

**Important:** Use care when selecting the **sshd\_config** file. There are two files, one for the server named **sshd\_config** and one for the client named **ssh\_config**.

- ▶ **ssh-agent** authentication agent (for storing private keys)  
**ssh-agent** is an authentication agent that can store private keys for use with public key authentication. This agent allows a user to load their private key into memory to avoid retying the pass phrase each time an SSH connection is started. It is typically started at the beginning of one session, and subsequent program calls are started as clients to the **ssh-agent** process.
- ▶ **ssh-add** (tool that adds keys to the **ssh-agent**)  
**ssh-add** tries to load private keys from private key files in a user's ~/.ssh directory. The file names that **ssh-add** is looking for are defined in the ssh\_config file.
- ▶ **sftp** (FTP-like program that works over the SSH1 and SSH2 protocol)  
**sftp** is a secure FTP replacement. As with all implementations of **sftp** on other platforms, **sftp** can only transfer data in binary format. **sftp** does not provide the enhanced functions that are available in the i5/OS FTP utility when transferring files in the QSYS.LIB file system. Nor does **sftp** provide the CCSID data conversion options available in the i5/OS FTP utility.
- ▶ **scp** file copy program  
**scp** is a secure file copy program and an alternative to **sftp** for copying a single file in the integrated file system. It is the OpenSSH version of **rcp**.
- ▶ **ssh-keygen** key generation tool  
**ssh-keygen** is a public and private key generation and management tool. **ssh** allows users to authenticate using these public and private keys as an alternative to using their operating system signon password. As the key names suggest, public keys can be freely distributed, while private keys should be protected.
  - Client side  
 The public key can be distributed to the server to which the client connects. The corresponding private key stays on the client and is typically stored encrypted via a pass-phrase key. Whenever the client user wants to use the private key for authentication, the user must enter the pass phrase to unlock the key.
  - Server side  
 The public key can be distributed to any client that connects to this server. The private key is stored in a file on the server. This file should be protected via object authorities. Usually, private key files on the server side are not encrypted using a pass phrase. If this is the case, somebody must enter the pass phrase every time the server daemon starts.  
 Under i5/OS, the keys are stored in files in the integrated file system and private key files provide no public access.

In addition to the utilities, such as **ssh**, **sftp**, and so forth, OpenSSH provides other functions, which include:

- ▶ **X11 forwarding**: Provides encryption of remote X Window System network traffic
- ▶ **Port forwarding**: Allows forwarding of TCP/IP connections to a remote system over an encrypted channel. This function is particularly useful for applications that do not support Secure Sockets Layer (SSL) encryption, such as Post Office Protocol (POP) or Simple Network Management Protocol (SNMP).
- ▶ **Data compression**: OpenSSH compresses data before it encrypts data using **zlib** for compression. This can improve the overall performance.

- ▶ **Kerberos and AFS® Ticket Passing:** Passes tickets for Kerberos and AFS on to the remote system. A user can access all Kerberos and AFS services without entering a password again.
- ▶ **Cryptographic functions:** Uses the OpenSSL cryptographic library

## 1.2 i5/OS implementation

IBM Portable Utilities for i5/OS, License Program Option (LPO) 5733-SC1, is available since February 2005 for V5R3 and now also for V5R4 i5/OS users. This LPO contains the OpenSSH, OpenSSL, and zlib open source packages ported to i5/OS using the i5/OS PASE (Portable Solutions Application Environment) runtime environment. The LPO requires a minimum of i5/OS V5R3 and requires that i5/OS Option 33 (i5/OS PASE) is installed.

Only a single English build is available. However this single build includes the following translations of the OpenSSH messages, which are used based on the LANG and NLSPATH environment variable settings:

- ▶ CA\_ES and ca\_ES (Catalan)
- ▶ CS\_CZ and cs\_CZ (Czech)
- ▶ DE\_DE and de\_DE (German)
- ▶ EN\_US and en\_US (English)
- ▶ ES\_ES and es\_ES (Spanish)
- ▶ FR\_FR and fr\_FR (French)
- ▶ HU\_HU and hu\_HU (Hungarian)
- ▶ IT\_IT and it\_IT (Italian)
- ▶ JA\_JP and ja\_JP and Ja\_JP (Japanese)
- ▶ KO\_KR and ko\_KR (Korean)
- ▶ PL\_PL and pl\_PL (Polish)
- ▶ PT\_BR and pt\_BR (Portuguese)
- ▶ RU\_RU and ru\_RU (Russian)
- ▶ SK\_SK and sk\_SK (Slovak)
- ▶ ZH\_CN and Zh\_CN and zh\_CN (Simplified Chinese)
- ▶ ZH\_TW and Zh\_TW and zh\_TW (Traditional Chinese)

The versions and installation directories for the products are as follows:

- ▶ The OpenSSH version is 3.5p1 and is located in the /QOpenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.5p1/ directory.
- ▶ The OpenSSL version is 0.9.7d and is located in the /QOpenSys/QIBM/ProdData/SC1/OpenSSL/openssl-0.9.7d/ directory.
- ▶ The zlib version is 1.1.4 and is located in the /QOpenSys/QIBM/ProdData/SC1/zlib/zlib-1.1.4/ directory.

### System-wide configuration settings

The **sshd** daemon (sshd\_config file) and **ssh** client (ssh\_config file) system-wide settings are stored in the integrated file system in directory /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc.

### User-specific configuration settings

Certain `ssh` client settings can be overridden or defined by the individual user. Each user who uses SSH must have a home directory, such as `/home/barlen`. Under this home directory, there is a hidden directory with the name `.ssh`, for example, `/home/barlen/.ssh`. This directory can contain a user's public/private key pairs, a config file, a `known_hosts` file, and an `authorized_keys` file.

## 1.3 Installing the IBM Portable Utilities for i5/OS license program

IBM Portable Utilities for i5/OS contains the OpenSSH, OpenSSL, and zlib open source products. The license program option number for Portable Utilities for i5/OS is 5733-SC1. Since the license program option is available in U.S. English only, you must restore the product as shown in the following two commands:

```
RSTLICPGM LICPGM(5733SC1) DEV(OPTxx) OPTION(*BASE) RSTOBJ(*ALL) LNG(2924)
RSTLICPGM LICPGM(5733SC1) DEV(OPTxx) OPTION(1) RSTOBJ(*PGM)
```

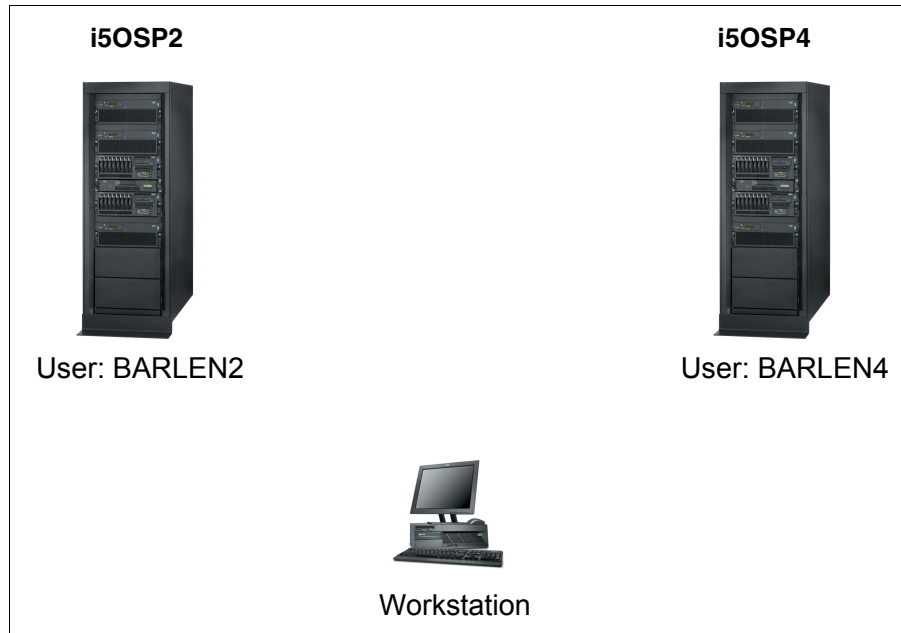
**Note:** Option 1 only contains program objects. In addition to product 5733-SC1, you must install i5/OS PASE, which is shipped as i5/OS option 33.

## 1.4 Redpaper example environment

Throughout the chapters of this Redpaper, the examples shown are based on the following scenario characteristics:

- ▶ The SSHD daemon is running on two i5/OS partitions with the host names `i5OSP4` and `i5OSP2`.
- ▶ SSH connections are established between the previously mentioned partitions.
- ▶ SSH connections are also established from i5/OS to an HMC.
- ▶ The user profiles that are used are `BARLEN2` for the `i5OSP2` system and `BARLEN4` for the `i5OSP4` system.
- ▶ Both i5/OS operating systems are at software level Version 5 Release 3 Modification 0.

Figure 1-1 illustrates our environment.



*Figure 1-1 Example environment*

## 1.5 Additional information

You can find additional information about SSH and Portable Utilities for i5/OS at the following Web addresses:

- ▶ OpenSSH  
<http://www.openssh.org/>
- ▶ IBM Portable Utilities for i5/OS  
<http://www.ibm.com/servers/enable/site/porting/tools/openssh.html>





## Setting up and running the sshd daemon

In this chapter, we explain how you set up the Secure Shell (SSH) server (**sshd** daemon). You must perform these same steps on every system that runs an **sshd** daemon.

For a client to establish an SSH connection to a server, such as the IBM eServer™ iSeries™ server, the **sshd** daemon must run on the server side. Before you can start the **sshd** daemon, you must perform some initial setup tasks.

The setup tasks are as follows:

- ▶ Set up public and private keys for SSH protocol 1 (rsa1 key).
- ▶ Set up public and private keys for SSH protocol 2 (rsa and dsa keys).
- ▶ If changes are required, modify the **sshd** configuration file.
- ▶ Start **sshd** or schedule autostart of **sshd**.

You must install the license program option IBM Portable Utilities for i5/OS (5733-SC1) prior to performing the setup explained in this chapter. In addition to 5733-SC1, you must have installed i5/OS PASE (Portable Solutions Application Environment) option 33.

## 2.1 Setting up the sshd daemon

Public key authentication is used when establishing an SSH session. At least the **sshd** daemon requires one or more public key pairs to support public key authentication. Different key types are used for the SSH1 and SSH2 protocols. Perform the following steps to generate the required key pairs for both SSH protocol types:

1. Sign on to your System i platform with a user profile that has \*ALLOBJ special authority.
2. From the command line, enter the following command to start an i5/OS PASE shell session:

```
CALL QP2TERM
```

3. In the i5/OS PASE shell session, run the following commands in the order shown:

```
cd /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc
ssh-keygen -t rsa1 -b 2048 -f ssh_host_key -N ''
ssh-keygen -t dsa -b 2048 -f ssh_host_dsa_key -N ''
ssh-keygen -t rsa -b 2048 -f ssh_host_rsa_key -N ''
```

These commands generate public and private key pairs for the SSH1 and SSH2 protocols. They are stored in the /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc directory. The -N parameter specifies the passphrase that is used to protect the private keys. The passphrase is used as a password to unlock the keys.

For a server, we recommend that you *do not specify a passphrase*, because the server usually starts in unattended mode. If a passphrase is specified, the server prompts the user to enter the passphrase. Therefore, an empty passphrase is specified.

**Note:** For the scenario in our Redpaper, we set up the **sshd** daemon environment up on both the i5OSP4 and i5OSP2 systems.

### 2.1.1 Modifying the sshd daemon system configuration

The /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc directory also contains a configuration file for the server (sshd\_config) and the client (ssh\_config). Customizing the **sshd** configuration file is optional. It is only required when the default values do not meet your requirements. As shown in Figure 2-1, the provided sshd\_config file contains all possible configuration directives. By default, these directives are commented out with the number sign (#).

```
#Port 22
#Protocol 2,1
#ListenAddress 0.0.0.0
#ListenAddress ::

# HostKey for protocol version 1
#HostKey /QOpenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.5p1/etc/ssh_host_key
# HostKeys for protocol version 2
#HostKey /QOpenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.5p1/etc/ssh_host_rsa_key
#HostKey /QOpenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.5p1/etc/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
#KeyRegenerationInterval 3600
#ServerKeyBits 768
```

Figure 2-1 The sshd\_config file

The values behind the keywords are the default values that the **sshd** daemon uses when the configuration file is not changed. You can edit the configuration file with this command:

```
EDTF ' /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc/sshd_config'
```

For more information about the configuration file and its directives, refer to the OpenSSH Web site at the following address:

<http://www.openssh.org>

## 2.2 Starting the sshd daemon with Submit Job (SBMJOB)

Eventually you must start the **sshd** daemon with the **sshd** shell command. You can also submit the job as a batch job as shown in the following example:

```
SBMJOB CMD(CALL PGM(QP2SHELL) PARM('/QOpenSys/usr/sbin/sshd'))
```

After you finish the configuration, you can start the **sshd** daemon. You can either start the daemon manually or put it into the startup program to start the daemon automatically when the system starts. The user under which the **sshd** daemon runs must have \*ALLOBJ special authority to login users via **ssh**. We recommend that you create a separate user just for running the **sshd** daemon. The user profile name must be eight characters or less.

## 2.3 Starting the sshd daemon in a dedicated subsystem environment

To better control the environment and resources that are used by SSH jobs, we recommend that you run SSH jobs in a dedicated subsystem. This becomes even more obvious when we discuss how the **sshd** environment works. When you start the **sshd** daemon in i5/OS, a single job for the daemon is started. When a client establishes an SSH session to the daemon, the daemon spawns a new job for this particular client. When a user is authenticated for this client session, another job is spawned. In addition, if the user runs a command or job, another job is started, which means, that you might end up with *three* jobs for a single client user. Therefore we recommend that you run all SSH jobs in a separate subsystem.

To set up the required subsystem environment, you must have at least the following i5/OS objects:

- ▶ Subsystem description (SBSD) with routing and memory entries and an autostart job entry
- ▶ Job queue (JOBQ)
- ▶ Job description (JOBQD)
- ▶ User profile (USRPRF): This object is recommended to run **sshd** under a dedicated user profile.

The following steps show an example of starting the **sshd** daemon in a simple subsystem environment:

1. Create a subsystem description using the following CL command:

```
CRTSBSD SBSD(SSHLIB/SSHSBS) POOLS((1 *BASE)) TEXT('SSH jobs subsystem')
```

This command creates a subsystem description SSHSBS in the SSSLIB library and a single memory pool is assigned. You might want to create a dedicated memory pool in your environment instead of using the system's base pool.

2. Create a job queue for submitting the job to the subsystem:

```
CRTJOBQ JOBQ(SSHLIB/SSHJOBQ) TEXT('SSH job queue')
```

3. Create a user profile for the daemon job:

```
CRTUSRPRF USRPRF(SSHDUSR) PASSWORD(*NONE) INLMNU(*SIGNOFF) LMTCPB(*YES) SPCAUT(*ALLOBJ)  
TEXT('SSHD Daemon user profile')
```

This user profile is used to run the **sshd** daemon, and therefore, should not be used to sign on to the system. To ensure this, create the profile without a password and specify **\*SIGNOFF** for the initial menu. In addition, set limited capabilities for the user profile to **\*YES**.

4. Create a job description for the subsystem autostart job entry:

```
CRTJOBQ JOBQ(SSHLIB/SSHJOBQ) JOBQ(SSHLIB/SSHJOBQ)  
TEXT('Job description for SSHD autostart') USER(SSHDUSR)  
RQSDTA('CALL PGM(QP2SHELL) PARM('/QOpenSys/usr/sbin/sshd')')
```

5. Create a class for the subsystem. The class defines the run priority of the **ssh** jobs and other resource related parameters.

```
CRTCLS CLS(SSHLIB/SSHCLS) TEXT('SSH job class')
```

6. Add a routing entry to the subsystem:

```
ADDRTGE SBSD(SSHLIB/SSHSBS) SEQNBR(1) CMPVAL(*ANY) PGM(QCMD) CLS(SSHLIB/SSHCLS)
```

For the autostart job entry to start the **sshd** daemon job, a routing entry is required in the job in the subsystem.

7. Add the job queue that you previously created to the subsystem description:

```
ADDJOBQE SBSD(SSHLIB/SSHSBS) JOBQ(SSHLIB/SSHJOBQ) MAXACT(*NOMAX) SEQNBR(10)
```

8. Add the autostart job entry to the subsystem description:

```
ADDAJE SBSD(SSHLIB/SSHSBS) JOB(SSHD) JOBQ(SSHLIB/SSHJOBQ)
```

Whenever the subsystem is started, the job, as specified in the autostart job entry through the job description, is started. It runs with the priority defined in the class **SSHCLS**. It also runs under the **SSHDUSR** user profile.

To fully automate the startup of the **sshd** daemon at IPL time, you must change your startup program to include the STRSBS SSHLIB/SSHSBS command. When the subsystem is started with the previously created subsystem environment and no SSH connection is established, you should see one job running as shown in Figure 2-2.

```

                                Work with Active Jobs                                I50SP4
                                                                                   04/05/06 17:37:28
CPU %:      1.0      Elapsed time: 00:00:02      Active jobs: 187

Type options, press Enter.
  2=Change  3=Hold  4=End  5=Work with  6=Release  7=Display message
  8=Work with spooled files 13=Disconnect ...

Opt  Subsystem/Job  User      Type  CPU %  Function      Status
     SSHSBS        QSYS      SBS    .0      DEQW
     SSHD         SSHDUSR  BCI   .0    PGM-sshd   SELW

                                                                                   Bottom

Parameters or command
===>
F3=Exit   F5=Refresh   F7=Find   F10=Restart statistics
F11=Display elapsed data  F12=Cancel  F23=More options  F24=More keys

```

Figure 2-2 SSH subsystem job

For more information about subsystems and work management in general, refer to the iSeries Information Center at the following Web address:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r3/index.jsp>





## Establishing an SSH session

In this chapter, we explain how to set up a user environment on i5/OS and how to establish a Secure Shell (SSH) session between two i5/OS partitions or systems.

## 3.1 Preparing the user environment

Certain one-time setup tasks must be performed for every user who will use SSH, regardless of whether the user establishes a session to i5/OS or initiates a session from an i5/OS PASE (Portable Solutions Application Environment) session. The major prerequisites for each user who uses SSH are summarized as follows:

- ▶ The user profile name can be a maximum of eight characters long. Therefore, the i5/OS supported maximum length of ten characters cannot be used for a user who uses SSH.
- ▶ Each user must have a home directory in the integrated file system.
- ▶ Certain authorities must be set for the home directory.

### 3.1.1 Creating the home directory

Each user profile that authenticates to the `sshd` daemon when establishing a session to i5/OS or initiating a session from within i5/OS to another system's `sshd` daemon, must have a home directory in the integrated file system. By default, when you create a user profile in i5/OS, the home directory `/home/usrprf` is set in the user profile. However, because it is not automatically created, you must create the directory manually as shown in the following example, where `usrprf` is the user profile name:

```
MKDIR '/home/usrprf'
```

In the scenario described in 1.4, "Redpaper example environment" on page 5, the user profile on the i5OSP4 system is BARLEN4. Therefore, the command to create the home directory is:

```
MKDIR '/home/barlen4'
```

We also create the home directory for user BARLEN2 on the system i5OSP2.

### 3.1.2 Setting the home directory permissions

It is always a good idea to limit access permissions to any object in i5/OS to what a user or the public really needs. In a default environment as shipped with i5/OS, the integrated file system root directory has `*RWX` public authorities, so has the `/home` directory. That means, when you create a home directory without specifying the public authority settings, every user's home directory will also have a public `*RWX` (equivalent to `*ALL`) authority. You must change this for SSH, especially when using public key authentication.

```
CHGAUT OBJ('/home/barlen4') USER(*PUBLIC) DTAAUT(*EXCLUDE) OBJAUT(*NONE)
```

You also need to change the authority settings for the primary group. This also applies when you do not have a primary group assigned yet. The reason is that integrated file system keeps the default primary group authorities in the background. They can be displayed in the Qshell or i5/OS PASE shell with the command `ls -la /home/barlen4`. In this case, run the following CL command to set the permissions correctly:

```
CHGPGP OBJ('/home/barlen4') NEWPGP(*NONE) DTAAUT(*RX)
```

**Note:** The equivalent shell (Qshell or i5/OS PASE shell) command to set the public (other) and group authorities is:

```
chmod 750 /home/barlen4
```

We also set the permissions for the home directory for user BARLEN2 on system i5OSP2.



## 3.2 Using SSH between i5/OS environments

As mentioned in 1.1, “OpenSSH tools and files” on page 2, the **ssh** utility is a client utility that provides access to a remote system’s UNIX-type shell. In the case of an i5/OS environment, an SSH session is established to the i5/OS PASE shell environment. i5/OS PASE is the AIX® runtime environment on i5/OS. There are special considerations that you must attend to when initiating an SSH session from the i5/OS PASE shell. In this section, you learn how to establish an SSH session between the i5OSP4 (client) system and the i5OSP2 (**sshd**) system.

**Note:** The special program switches that you use in the following steps only apply when you initiate an SSH session from the i5/OS PASE shell environment. If you establish a session from Linux or Windows to the i5/OS PASE environment, the special parameter is not required.

Perform the following steps to establish an SSH session from the i5OSP4 system to the i5OSP2 system.

1. Start an i5/OS PASE shell session by entering the following command:

```
CALL QP2TERM
```

**Tip:** You might want to familiarize yourself with the **ssh** shell command and its various parameters. There is a significant number of parameters to choose from. You can override most of the system-wide settings for the **ssh** client environment, such as the protocol version. Use the following command to see all parameter options for the **ssh** program:

```
ssh -?
```

2. Establish an SSH session to the i5OSP2 system by entering the following command:

```
ssh -T barlen2@i5OSP2
```

This command establishes an SSH session and tries to sign on with user BARLEN2 to the i5OSP2 system. The **-T** switch is important when you initiate a session from the i5/OS PASE shell. It causes the **ssh** program not to try to allocate a TTY device. This is special to the i5/OS PASE environment. Without specifying the switch (parameter), you receive an error message that the system call received a parameter that is not valid and the connection is closed. The reason for this is that an i5/OS terminal session does not represent a true TTY terminal as all UNIX-type terminals do.

**Note:** You do not have to specify a user profile name with the **ssh** command. If omitted, the **ssh** command tries to log in with the user profile name that is used on the source system.

**Important:** If you set up the **sshd** (server) environment and establish an SSH session with your user profile for the first time, you see the following message indicating that the host key verification failed. The remote system's public key is not known on the source system.

```
ssh i5OSP2
The authenticity of host 'i5osp2 (172.17.17.29)' can't be established.
. key fingerprint is RSA.
Are you sure you want to continue connecting (yes/no)?
no
Host key verification failed.
$
```

When you answer “yes”, the remote systems public key is permanently added to your `known_hosts` file on the source system in the `.ssh` directory in the user's home directory. When the remote system's public key is in your `known_hosts` file, subsequent **ssh** requests from the same source user to the same target system no longer issue the message.

When asked whether you want to continue connecting, answer “yes” to add the remote host's public key to the `known_hosts` file.

3. When prompted to enter your password, enter the password of user **BARLEN2** on the **i5OSP2** system. You should authenticate successfully and end up in the **i5/OS PASE** shell of the **i5OSP2** system. Like in any other UNIX-type environment, you do not see any message when a command completes successfully, but rather only when an error occurs.

You can now issue any shell command. This command runs on the **i5OSP2** system. In Figure 3-1, the **pwd** (display the current directory) and **hostname** (display the IP host name) commands are issued after they are successfully authenticated.

```
$
> ssh -T barlen2@i5OSP2
The authenticity of host 'i5osp2 (172.17.17.29)' can't be established.
. key fingerprint is RSA.
Are you sure you want to continue connecting (yes/no)?
> yes
Warning: Permanently added 'i5osp2,172.17.17.29' (RSA) to the list of known hosts.
barlen2@i5osp2's password:
> pwd
/home/barlen2
> hostname
i5osp2.stgt.spc.ihost.com
```

*Figure 3-1 Running the **pwd** and **hostname** commands on the **i5OSP2** system*

4. Enter the **exit** command to return to your source system (**i5OSP4**) and close the SSH connection.
5. As mentioned previously, the **sshd** daemon's public key of the target system is stored in the `known_hosts` file on the source system. This allows the client to verify the public key of the target server the next time the client establishes the connection. The `known_hosts` file is stored in the client's user home directory in the subdirectory `.ssh`. The `.ssh` directory is by default a hidden directory. When using the Work with Object Links (WRKLNK) CL command to display the contents of the user's home directory, it appears to be empty, but that is not true.

You can enter the following CL command to display the hidden files and directories in a 5250 session:

```
WRKLNK OBJ('/home/barlen4/*') DSPOPT(*ALL)
```

As shown in Figure 3-2, DSPOPT \*ALL also shows the hidden .ssh directory.

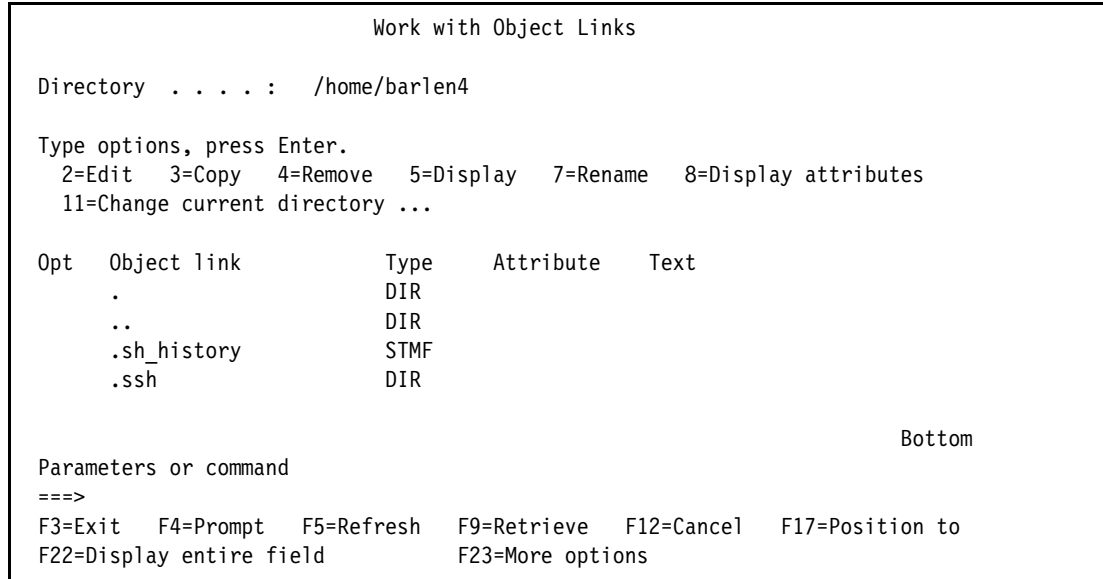


Figure 3-2 WRKLNK command output with the DSPOPT set to \*ALL

When using option 5 to go into the .ssh subdirectory, you see the known\_hosts file.

**Tip:** In a perfect world, every function that you perform on a system runs without any problems and always completes successfully. Unfortunately, this is not always the case. Therefore, it is sometimes important to know where a problem might come from. The **ssh** program includes a switch to provide extensive logging, which you can specify when starting an SSH session. Enter the following command to enable debug logging for **ssh**.

```
ssh -Tv barlen2@i5osp2
```

The **-v** switch enables logging. You can specify up to three **v** characters to increase verbosity.

### 3.2.1 Using the ssh utility to run commands remotely

The **ssh** utility has several switches and parameters that allow you to run a shell command remotely without first entering the remote shell. The following examples show you a couple of different ways to use this function:

- The following command establishes an SSH session under user profile BARLEN2 to the i5OSP2 system and submits a backup job in i5/OS:

```
ssh -T barlen2@i5osp2 'system "SBMJOB CMD(CALL PGM(TOOLS/BACKUPLIBS) PARM(*ALL))
JOB(BACKUP)" '
```

As a result of this command, the shell displays the following i5/OS message that was issued on the remote system:

```
CPC1221: Job 015242/BARLEN2/BACKUP submitted to job queue QBATCH in library QGPL.
```

The **system** command is used in the i5/OS PASE shell to run i5/OS commands.

- In this example, the Display Message (DSPMSG) command for displaying the QSYSOPR message queue is run:

```
ssh -T barlen2@i5osp2 'system "DSPMSG QSYSOPR" '
```

As opposed to the first example where you received the message about the submission of the job, the DSPMSG command returns the output to the source system's i5/OS PASE shell.

```
CPC9802: Printer output created.
5722SS1 V5R3M0 040528           Messages in queue - QSYSOPR
Page 0001
MSGID SEV MSG TYPE
CPI1E23 00 COMPLETION Cleanup has started.
QSYSSCD QPGMR 012895 QEZEVLHL 0000 01/05/06 23:00:21.246696 QPGMR
CPI1E88 00 INFO Cleanup of OfficeVision/400 calendar items started.
QCLNCALITM QPGMR 014930 QEZCLOFC 0000 01/05/06 23:00:21.273736 QPGMR
```

**Note:** These two examples always require you to enter a password when connecting to the remote system. In many cases, this is not desirable, because some commands should be run in unattended or batch mode. Automatic signon can be achieved with **ssh** by using public key authentication as described in 4.2.1, “Running the scp command in batch mode” on page 27.

## 3.3 Using SSH from other platforms to i5/OS

Since OpenSSH is available for many different system platforms, you can establish an **ssh**, **scp**, or **sftp** session between other systems and i5/OS. In Linux, AIX, or any other UNIX-type platform, the standard commands (**ssh**, **scp**, **sftp**) are available. On a Windows platform, you can use a free **ssh** utility called PuTTY, which can be downloaded from the Internet at the following Web address:

<http://www.putty.nl/download.html>

In the following section, you establish a basic SSH connection from a PuTTY **ssh** client to the i5/OS **sshd** daemon.

### 3.3.1 Using PuTTY to establish an SSH connection to i5/OS

The PuTTY program is an **ssh** client with a graphical user interface that is used often in a Windows environment. In addition to the PuTTY **ssh** client program, several other utilities are available that are used together with PuTTY (for example, **pageant** and **puttygen**) or that use the SSH protocol for file transfer with PSCP or PSFTP.

The following steps show you how to establish an SSH session with PuTTY to the i5/OS SSHD daemon. These steps presume that you have installed PuTTY on your workstation.

**Note:** Remember to set up the i5/OS user environment as described in 3.1, “Preparing the user environment” on page 14, before you continue with the following steps.

1. Start the PuTTY client. When you start it without parameters, the graphical interface starts.
2. Enter the following connection information:
  - a. For Host Name (or IP address), type **i5OSP4**.
  - b. For Port, type **22**.
  - c. For Protocol, select **SSH**.

These are the only parameters that you need to specify to connect via **ssh** to an **sshd** daemon.

3. In the Category pane, click **Connection** → **SSH**. Familiarize yourself with the connection options for the SSH protocol. In the SSH properties section, you can define the SSH protocol that you want the client to use and the encryption algorithms. You can also select whether to use zlib compression.
4. In the Category pane, click **Session**, to return to the session definition section.
5. To reuse the connection information for later use, you can save the settings under profile. In the Saved Sessions field, enter the profile description SSH to i5OSP4. Then click **Save** to save your session connection details.

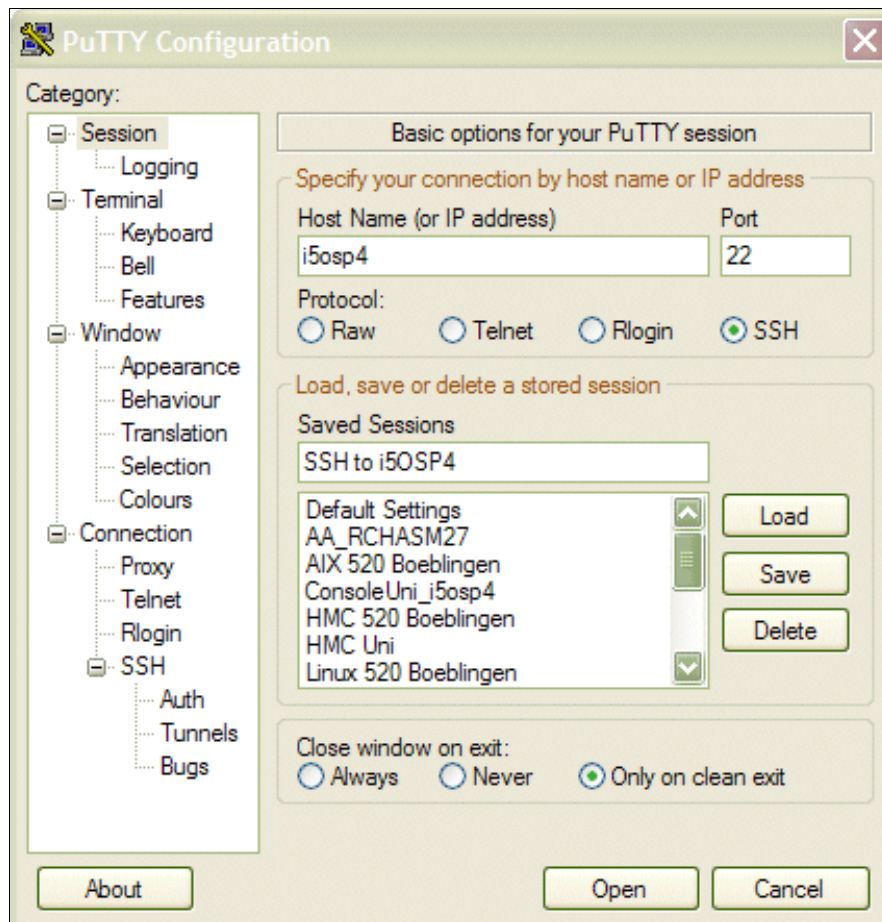


Figure 3-3 PuTTY session details i5OSP4

6. Leave the PuTTY client open.

## Establishing an SSH session

Now it is time to test the SSH connection. Since **ssh** is a Secure Shell that originated from the UNIX environment, an SSH session to i5/OS takes you into an i5/OS PASE shell.

1. In the open PuTTY client, click **Open**. Then the **ssh** client connects to the **sshd** server.
2. When you connect the first time to the **sshd** server, a warning message is displayed as shown in Figure 3-4.



Figure 3-4 The ssh client warning message

**ssh** uses public key authentication during session establishment. Initially, only server authentication is performed. That is, the client authenticates the server. This involves the server sending its public key to the client. If you connect the first time, the client does not know anything about the server's public key. Therefore, it displays a message indicating that the client is not aware of this server. You choose to trust the presented key and continue without (select No) storing the key for future sessions or select Yes to store the key on the client. When you select Yes and establish in the future another session to the server, the client already has the key and trusts the server connection, and no longer displays any message.

**Note:** When you change the server's public and private key pair for **sshd**, you see the warning message again.

Select **Yes** to store the server's public key the client. With PuTTY, the key is stored in the Windows registry.

3. At the login prompt, log in with your i5/OS user credentials. In this example, we login as **barlen4**. You are then prompted for your i5/OS user profile password. You are then connected to the i5/OS PASE shell on the i5OSP4 system.

```
login as: barlen4
barlen4@i5osp4's password:
$
```

Figure 3-5 The ssh shell session

4. In the SSH session, enter the **pwd** command to check your current directory settings. You should be in your **/home/barlen4** directory.
5. To perform another test, you might want to run a command, such as **system dspneta,pwd** to display the i5/OS network attributes in your shell session.



## File transfer and public key authentication with OpenSSH

In addition to an interactive **ssh** shell session, there are two programs for transferring files. The first one is the **scp** program, which allows you transfer a single file from the command line to another system. The second program is called **sftp**, which is similar to regular File Transfer Protocol (FTP). It starts a shell where you can enter FTP subcommands.

The **scp** and **sftp** programs can be used from any system that supports Secure Shell (SSH). However, when you try to use these programs from the i5/OS PASE (Portable Solutions Application Environment) shell, you run into the same problem as with **ssh**, namely that the i5/OS PASE shell session does not represent a true TTY terminal.

The major difference between **ssh** and **scp** and **sftp** is that the **scp** and **sftp** programs do not support the -T switch (do not allocate a TTY terminal). The only way to use **scp** and **sftp** from an i5/OS PASE shell session is by using public key authentication instead of password authentication. This, of course, has a significant advantage. In many System i accounts, batch FTP is used to transfer files from one System i platform to another one in unattended mode. This requires that the FTP commands are provided in a source file, which includes the password in clear text. Now, with **scp** and public key authentication, you can transfer files without potentially exposing passwords.

## 4.1 Setting up public key authentication

An alternative to password authentication is the use of public key authentication. This type of authentication involves the use of a private and public key pair. The public key can be distributed freely, but the private key should always be kept protected by the owner of the key.

The setup for public key authentication requires the following steps:

1. Create a private and public key pair on the source system for your user profile.
2. Transfer the public key to the destination system. This is the system to which you want to establish an **scp** or **sftp** connection. If you establish **ssh** (including **scp** and **sftp**) sessions to multiple systems, you must transfer the key to every destination system to which you want to connect.
3. Append the public key to the `authorized_keys` file on the destination system or systems. The `authorized_keys` file resides in the `/home/~user/.ssh` directory on the destination system. Only source users whose public key is stored in the `authorized_keys` file on the destination system can use public key authentication. The public key must correspond to the appropriate private key.

**Note:** You can store a source user's public key in multiple `authorized_keys` files on the target system for different target user profiles. That way a source user can establish a session to a target system by using several different user profiles on the target system.

In this scenario, you set up the following environment:

- ▶ You create a public and private key pair for user `BARLEN4` on the source system `i5OSP4`.
- ▶ You transfer the public key file of user `BARLEN4` from the `i5OSP4` system to the `i5OSP2` system and append the public key in the `BARLEN2` user profile's `authorized_keys` file.
- ▶ You enable source user `BARLEN4` to establish an **scp** session from the `i5OSP4` system to the `i5OSP2` system by using user profile `BARLEN2` on the target system.

Perform the steps in the following sections to set up public key authentication for the scenario environment.

### Generating the key pair

To generate the key pair, follow these steps:

1. If you have not already done so, sign on to the `i5OSP4` system with the user `BARLEN4`.
2. Enter the `i5/OS PASE` shell with the command:

```
CALL QP2TERM
```

3. The user who wants to create the key pair must be in the user's home directory. You can check this by entering the **pwd** command:

```
> pwd
/home/BARLEN4
$
```



4. In the i5/OS PASE shell, enter the following command to create a private and public key pair for the SSH1 protocol.

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

As shown in the command output (Figure 4-1), when prompted to enter a file name for the key, press Enter to accept the default location. The command generates the key pair and then asks for a passphrase to protect the key. Enter a passphrase and press Enter. The private key is stored in the `id_rsa` file. To use this key at a later time, you must provide the passphrase to decrypt the key file.

```
> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/BARLEN4/.ssh/id_rsa):
>
Enter passphrase (empty for no passphrase): Enter same passphrase again:
Your identification has been saved in /home/BARLEN4/.ssh/id_rsa.
Your public key has been saved in /home/BARLEN4/.ssh/id_rsa.pub.
The key fingerprint is:
f9:76:31:f4:e7:22:24:31:c2:b8:8f:13:58:5a:9d:afbarlen4@i5osp4.stgt.spc.ihost.com
$
```

Figure 4-1 The `ssh-keygen` command output

The command creates two files in the `.ssh` directory:

- **id\_rsa**: This file contains the private key.
- **id\_rsa.pub**: This file contains the corresponding public key.

**Note:** If you want to use `ssh`, `scp`, or `sftp` in unattended mode, for example, in a batch job, we recommend that you do not use a passphrase, especially when running that job in the night when nobody is there to enter the passphrase. In this case, you press Enter, without typing a passphrase. Without a passphrase, the only protection of the private key is the object level permission to the file itself. By default, the private key file is created with an object level permission of `*EXCLUDE` for group and public authorities.

5. Press F3 to exit the i5/OS PASE shell session.

## Transferring the public key

Now that the key pair is created, in the next steps, you transfer the public key file to the remote system where you want to connect to via `ssh`, `scp`, or `sftp`.

1. Send the `id_rsa.pub` file to the remote i5OSP2 system using your user credentials on i5OSP2 (BARLEN2). Enter the following FTP command to start a sub-shell FTP window:

```
FTP RMTSYS(i5OSP2)
```

2. Use the FTP session commands as shown in Figure 4-2. After the transfer is completed successfully, exit the FTP session by entering the **quit** command.

```
Previous FTP subcommands and messages:
Connecting to host I5OSP2 at address 172.17.17.29 using port 21.
220-QTCP at i5osp2.stgt.spc.ihost.com.
220 Connection will close if idle more than 5 minutes.
> barlen2
331 Enter password.
230 BARLEN2 logged on.
    OS/400 is the remote operating system. The TCP/IP version is "V5R3M0".
250 Now using naming format "0".
257 "QGPL" is current library.
> bin
200 Representation type is binary IMAGE.
> nam 1
250 Now using naming format "1".
Server NAMEFMT is 1.
Client NAMEFMT is 1.
> put /home/barlen4/.ssh/id_rsa.pub /home/barlen2/id_rsa.pub
227 Entering Passive Mode (172,17,17,29,44,152).
150 Sending file to /home/barlen2/id_rsa.pub
250 File transfer completed successfully.
    243 bytes transferred in 0.019 seconds. Transfer rate 13.096 KB/sec.
> quit
```

Figure 4-2 FTP subcommands to transfer *id\_rsa.pub* from *i5OSP4* to *i5OSP2*

3. Keep the 5250 session on *i5OSP4* open.

## Adding the public key to the `authorized_keys` file

As mentioned in the introduction of this chapter, the `authorized_keys` file resides in the user's `.ssh` directory under the user's home directory. By default, the `.ssh` directory does not exist until you create it manually or start an SSH session from the *i5/OS PASE* shell to another system. In this scenario, you create the directory structure and set the necessary permissions manually. This time, all the steps are performed with *i5/OS PASE* shell commands instead of *i5/OS CL* commands.

1. Using a 5250 session, sign on to *i5OSP2* (your target system) with the user profile **BARLEN2**.

2. Enter the *i5/OS PASE* shell with the command:

```
CALL QP2TERM
```

3. Enter the following command to create the `.ssh` subdirectory. The home directory, along with the required authorities, was created previously.

```
mkdir /home/barlen2/.ssh
```

4. Set the required authorities for the `.ssh` directory:

```
chmod 700 /home/barlen2/.ssh
```

The numbers that follow the **chmod** command represent the authority bits in a UNIX-type file system. In this example, the value 700 indicates:

- The owner of the directory has Read, Write, and Execute permissions.
- The primary group of the directory has no access permissions.
- Other users have no access permissions for this directory.

**Note:** It is not required to remove the read (R) and execute (X) permission for Other and the Group. The only requirement is to remove the write (W) authority. However, it is always good to restrict access to security-sensitive information.

5. Enter the following shell command to verify the existence of the .ssh directory and the authority settings:

```
ls -al /home/barlen2
```

Figure 4-3 shows the output of the command.

```
> ls -al /home/barlen2
total 80
drwxr-s--- 3 barlen2 0          8192 May 05 15:38 .
drwxrwsrwx 3 qsys    0          8192 May 05 10:01 ..
-rw----- 1 barlen2 0           132 May 05 15:49 .sh_history
drwx--S--- 2 barlen2 0          8192 May 05 15:38 .ssh
-rwxr-x--- 1 barlen2 0          243 May 05 15:29 id_rsa.pub
$
```

Figure 4-3 The output of the `ls -la` command

6. In the i5/OS PASE shell, change the current directory to the .ssh directory with the following command:

```
cd /home/barlen2/.ssh
```

7. Append the public key file that you transferred to the i5OSP2 system to the `authorized_keys` file. Enter the following command to append the public key file:

```
cat /home/barlen2/id_rsa.pub >> authorized_keys
```

This command outputs the contents of the `id_rsa.pub` file and appends them to the existing `authorized_keys` file or creates a new `authorized_keys` file if it does not exist. Do *not* attempt to use the Edit File (EDTF) command or another editor to do this. The key in the file is long, and control characters or typos render the `authorized_keys` file useless.

8. Enter the command `ls -l` again to display the permissions of the `authorized_keys` file (see Figure 4-4).

```
-rw-rw-rw- 1 barlen2 0          243 May 05 15:53 authorized_keys
```

Figure 4-4 The `authorized_keys` file permissions after creation

The `authorized_keys` file must also have the write authorities removed. Otherwise public key authentication fails.

9. The `authorized_keys` file is a sensitive file. You do not want anybody to add their own keys to your file. Even better, you do not want anybody to see which keys you have in there. Therefore, you must set permissions accordingly with the following command:

```
chmod go-rwx authorized_keys
```

The command removes the read, write, and execute permissions from the group and other users. Figure 4-5 shows the result.

```
-rw----- 1 barlen2 0          243 May 05 15:53 authorized_keys
```

Figure 4-5 The `authorized_keys` file permissions after permission change

10. You can display the contents of the `authorized_keys` file with the following command:

```
cat authorized_keys
```

Figure 4-6 shows the output.

```
> cat authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA10ssSVdT7WUt5wN1RgSo6xw9Z9jrq46rQ8kgYXr139LCkSns1HLhGdC4GGmx
ZjUHV1MKbDnNUD99Cbzq/qa9TnYCSbx/u
RW56Mp0cFJvnFPDnLZW/6vYyzGXAfrn85hR56HZ0RZtfhDZWbiypA2If34SbjS6YtBfX8EHPjBmp2M=
barlen4@i5osp4.stgt.spc.ihost.com
$
```

Figure 4-6 The `authorized_keys` file after import of public key

For every source user who wants to authenticate with public key authentication using this target system's user profile, you must import (append) the corresponding public key.

**Note:** The most important requirement regarding the permissions for your home directory, the `.ssh` directory, and the `authorized_keys` files is that they are not group writable.

## 4.2 Using public key authentication with `scp` to transfer files

Now that you have created your key pair for key authentication, you must test whether you did everything correctly. If you want to use the same i5/OS PASE shell session for transferring multiple files with `scp`, you can store the private key in memory before using `scp`. To place the private key in memory, you must run the `ssh-agent` and `ssh-add` programs. The `ssh-agent` program allows you to have private keys stored in memory. The `ssh-add` utility is the actual tool that adds the keys to the agent.

First, let us work with the `ssh-agent` and `ssh-add` utilities.

1. You should still have the 5250 session on i5OSP4 open. In this scenario environment, as described in 1.4, "Redpaper example environment" on page 5, the session is still open with the user BARLEN4 signed on.

2. Start an i5/OS PASE shell session with the following CL command:

```
CALL QP2TERM
```

3. Enter the following command to start the `ssh-agent` program:

```
ssh-agent $SHELL
```

The parameter `$SHELL` refers to an environment variable that contains the path and name of the current shell (`/QOpenSys/usr/bin/sh`). The agent then runs in a new shell. Note that the `$SHELL` value is case sensitive. If the start of the agent is successful, you do not see any messages and no command prompt character is displayed.

4. Enter the following command to add your private key to the agents memory:

```
ssh-add
```

The `ssh-add` command tries to read all private key files from the user's `.ssh` directory.

**Note:** The default file names that **ssh-add** is looking for are defined in the system-wide ssh client configuration file /QOpenSys/QIBM/UserData/SC1/OpenSSH/openssh-3.5p1/etc/ssh\_config. This file contains the following directives:

```
# IdentityFile ~/.ssh/identity
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
```

The number sign (#) means that the directives show the default values that the system uses unless otherwise specified. If you want to override the values with your own values, you must remove the # sign.

The **ssh-add** program asks you for the passphrase that you used to protect the private key file. This is the passphrase that you entered when you created the key pair in step 4 on page 23.

Enter the passphrase. You should see a result similar to the one in Figure 4-7.

```
$
> ssh-agent $SHELL
> ssh-add
Enter passphrase for /home/BARLEN4/.ssh/id_rsa:
Identity added: /home/BARLEN4/.ssh/id_rsa (/home/BARLEN4/.ssh/id_rsa)
```

Figure 4-7 The ssh-add command

You are ready to use **scp** with public key authentication.

5. Let us assume that you want to transfer the ssh\_config file from the /home/sshdusr directory on the i5OSP4 system to the home directory of user BARLEN2 on the i5OSP2 system.

```
scp /home/sshdusr/ssh_config barlen2@i5OSP2:/home/barlen2/ssh_config
```

Notice the syntax of the **scp** command. The first parameter identifies the source file followed by a remote user at a specific system followed by the remote file name. In the case of a successful completion, the **scp** command, as all other commands, does not return any success messages. Only error messages are shown if an error should occur. If you omit the target user name, **scp** tries to use the same user profile as on the source system.

### 4.2.1 Running the scp command in batch mode

The steps in the previous section work fine in an interactive session mode. A user can enter several **scp** commands after loading the private key into the user environment with the **ssh-agent** and **ssh-add** utilities. But this process does not work well in a batch environment. Suppose you start a CL program in batch that transfers three files during the night. The program must submit an i5/OS PASE shell batch job with the program QP2SHELL. However, in a batch environment, no user who enters the passphrase is able to use the private key.

The answer to this problem might be to use an i5/OS PASE shell script that is started from a CL batch job or a CL program that calls the **scp** utility. The following section shows you how to set up an environment to transfer files with public key authentication in batch mode. The characteristics of the scenario are as follows:

- ▶ The user BARLEN4 is signed on to the source system i5OSP4.
- ▶ The user BARLEN2 is signed on to the target system i5OSP2.

- ▶ A file test1 should be transferred via **scp** from the source to the target system. The file is stored in the /home/barlen4 directory on the source system and should be transferred to the /home/barlen2 directory on the target system.
- ▶ Public key authentication should be used for the file transfer.
- ▶ The transfer should be started in CL batch job on the source system.

The following steps guide you through the process of creating the previously described environment.

## Setting up the public and private keys

The public and private key pair that was created in 4.1, “Setting up public key authentication” on page 22, uses a passphrase to protect the private key file. Whenever a user or job wants to use public key authentication and accesses the private key file, someone must enter the passphrase to *unlock* the key. This is not usable in a batch environment where nobody is available to enter a passphrase.

The problem can be solved by creating the private key without a passphrase. That way, the private key is stored in cleartext in the key file and can be used by any user or program that has access to the private key file. Therefore it is of utmost importance to properly secure your private key file. In the batch file transfer scenario, only the user profile, under which the batch job runs, should have read access to the key file. Public authority should be set to \*EXCLUDE.

**Tip:** Using public key authentication with **scp** in a batch environment has a significant advantage over traditional batch FTP in i5/OS. No passwords need to be hardcoded in cleartext in programs or files.

Perform the following steps to create the key pair for the batch environment:

1. On the source system i5OSP4, sign on via a 5250 session under user BARLEN4.
2. Start an i5/OS PASE shell session with the CL command:  

```
CALL QP2TERM
```
3. The user who wants to create the key pair needs to be in the user's home directory. This can be checked by entering the **pwd** command.  

```
> pwd
/home/BARLEN4
$
```
4. Change to the directory .ssh:  

```
cd .ssh
```
5. In the i5/OS PASE shell, enter the following command to create a private and public key pair for the SSH1 protocol. This time, the file name (-f) and passphrase (-N) parameters are specified as parameters on the **ssh-keygen** command. Also an RSA key is used for protocol SSH2.  

```
ssh-keygen -t rsa -f id_rsa -N ''
```

The private key is stored in the `id_rsa` file without passphrase protection. If a key already exists, you are prompted to override the existing key. In our scenario, the private key file already existed from the previous setup as shown in Figure 4-8.

```
> ssh-keygen -t rsa1 -f id_rsa -N ''
Generating public/private rsa1 key pair.
id_rsa already exists.
Overwrite (yes/no)?
> yes
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
d5:5e:06:c0:4c:db:2f:d3:f5:37:d2:e5:8c:34:77:46 barlen4@i5osp4.stgt.spc.ihost.com
$
```

Figure 4-8 Key generation without passphrase protection

6. You must transfer the `id_rsa.pub` public key file to the target system as described in “Transferring the public key” on page 23. You must also add this file to the target user’s authorized keys file as explained in the “Adding the public key to the authorized\_keys file” on page 24.

## Creating the CL batch program

You can run the `scp` file transfer from a CL program. The CL source shown in Figure 4-9 calls the i5/OS PASE shell batch environment and copies the `/home/barlen4/test1` file, on the i5OSP4 system, to the `/home/barlen2/test1` file on the i5OSP2 system.

```
/* ***** */
/* XFER - CL program to transfer file test1 from one system to another */
/*      system using the OpenSSH utility scp. */
/* ***** */
/* Written by: Thomas Barlen, IBM Germany, May 2006 */
/* ***** */

        PGM
        CALL      PGM(QP2SHELL) +
                  PARM(' /QOpenSys/QIBM/ProdData/SC1/OpenSSH/o+
                  penssh-3.5p1/bin/scp' '-B' +
                  '-i /home/barlen4/.ssh/id_rsa' +
                  '/home/barlen4/test1' +
                  'barlen2@i5osp2:/home/barlen2/test1')

        ENDPGM
```

Figure 4-9 CL source for calling `scp` from CL

The parameters for `scp` can also be passed as variables to the CL program. In the example shown in Figure 4-9, the `scp` utility is called through the QP2SHELL program from the integrated file system directory `/QOpenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.5p1/bin`. The `-B` parameter tells `scp` not to prompt for any user input. The name of the private key file that should be used for authentication is defined on the `-i` parameter. The two parameters that follow define the source file and the target file preceded with the target system’s user and system name.

You can then submit this program to batch by using the Submit Job (SBMJOB) CL command.

## Creating the i5/OS PASE shell script

Submitting several i5/OS PASE shell commands in a CL program runs each shell command in a different shell environment. To run several shell commands in one i5/OS PASE shell environment, a shell script can be used. In this example, a shell script with the name xfer is created in the user's home directory /home/barlen4.

1. Sign on with user BARLEN4 on the i5OSP4 system.
2. Enter the following command to edit the file xfer in the user's home directory:  
edtf '/home/barlen4/xfer'
3. Within the editor window, press F15 to open the Editor Options Screen.
4. In the EDTF Options Screen, select option 3 and specify an ASCII code page, such as 850, for the CCSID of the file and press Enter. Then enter option 5 and specify \*LF as the end of line (EOL) option. Finally press F12 to return to the editor window.

EDTF Options Screen			
Selection . . . . .	3		
1. Copy from stream file . . . . .	/home/barlen4/xfer		
2. Copy from database file . . . . .		Name	
Library . . . . .		Name, *LIBL, *CURL	
Member . . . . .		Name, *FIRST	
3. Change CCSID of file . . . . .	00850	Job CCSID: 00037	
4. Change CCSID of line . . . . .	*NONE		
5. Stream file EOL option . . . . .	*LF	*CR, *LF, *CRLF, *LF CR, *USRDFN	
User defined. . . . .		Hexadecimal value	

Figure 4-10 EDTF Options Screen

5. Create a shell script, starting with the shell script header as shown in Figure 4-11:

```
#!/bin/ksh
# Written by: Thomas Barlen, IBM Germany, May 2006
#
# ABSTRACT: Transfer files via scp using public key authentication
#
# Export the path where the OpenSSH utilities are stored:
export PATH=$PATH:/QOpenSys/QIBM/ProdData/SC1/OpenSSH/openssh-3.5p1/bin
# Run the scp commands:
scp -B -i /home/barlen4/.ssh/id_rsa /home/barlen4/test1
barlen2@i5osp2:/home/barlen2/test1
scp -B -i /home/barlen4/.ssh/id_rsa /home/barlen4/test2
barlen2@i5osp2:/home/barlen2/test2
scp -B -i /home/barlen4/.ssh/id_rsa /home/barlen4/test3
barlen2@i5osp2:/home/barlen2/test3
```

Figure 4-11 Creating a shell script



These shell script commands copy three files via **scp** from the source to the target system. All three commands run in the same shell session.

6. Set the authorities to the shell script to only allow authorized users to execute it. Enter the following commands in the order shown:

```
CHGAUT OBJ('/home/barlen4/xfer') USER(*PUBLIC) DTAAUT(*EXCLUDE) OBJAUT(*NONE)
CHGPGP OBJ('/home/barlen4/xfer') NEWPGP(*NONE) DTAAUT(*EXCLUDE) OBJAUT(*NONE)
```

7. Make sure that the Execute bit is set for the shell script. You can set it for user BARLEN4 with the command:

```
CHGAUT OBJ('/home/barlen4/xfer') USER(BARLEN4) DTAAUT(*RWX)
```

8. You can call this shell script from a CL program by using the following CL command:

```
CALL          PGM(QP2SHELL) PARM('/home/barlen4/xfer')
```

You can also pass the file names and other values as parameters or environment variables to the shell script.

## 4.3 Exploiting public key authentication with ssh

Public key authentication can be used with more than just **scp** or **sftp**. It can also be used with **ssh**. It allows you, for example, to submit remote commands through **ssh**. You can use the **ssh-agent** and **ssh-add** utilities as well as specify the private key file with the **-i** parameter on the **ssh** command to use public key authentication with **ssh**.





## Protecting traffic with SSH tunnels

A powerful function with `ssh` is the capability of tunneling or forwarding Internet Protocol (IP) traffic in a secure tunnel. This is useful for applications that do not support Secure Sockets Layer (SSL) or Transport Layer Security (TLS). An example of using a Secure Shell (SSH) tunnel might be if a user wants to use Telnet from one i5/OS to another i5/OS. Although the Telnet server in i5/OS supports SSL, the client does not. As an alternative to using virtual private network (VPN), you can establish an SSH tunnel between the two systems and then tunnel the Telnet traffic through the secure connection. Tunnels are established on a per port basis. That means, if you want to securely tunnel Post Office Protocol (POP) and Simple Mail Transfer Protocol (SMTP) traffic, you need two tunnels one for port 110 and one for port 25.

In this chapter, you set up SSH tunnels between two i5/OS environments and between a PC workstation and an i5/OS environment.

## 5.1 Setting up an SSH tunnel between i5/OS environments

In this section, you use public key authentication to establish a secure SSH tunnel between i5OSP4 and i5OSP2. The tunnel is used to establish secure 5250 Telnet sessions. Figure 5-1 illustrates the environment that you are going to use.

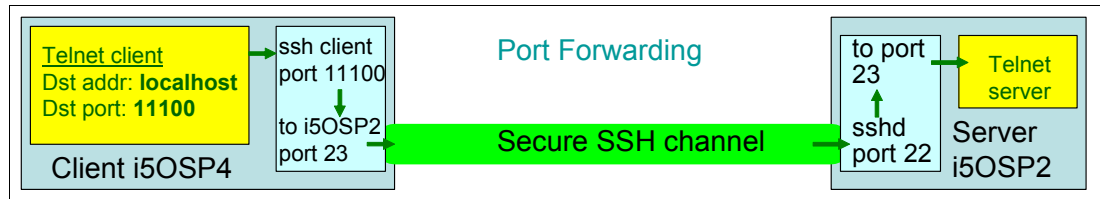


Figure 5-1 i5/OS SSH tunnel environment

A tunnel is set up on the client system. The *tunnel* is a server process that listens on a port. In this scenario, you use a port that is not being used by another job. We have chosen port number 11100 on the client side of the tunnel. The tunnel listens on the localhost address and ends at the remote systems **sshd** daemon. The daemon routes the traffic to port 23 on the remote system. This allows you to tunnel an otherwise unprotected 5250 session securely through an SSH tunnel. The only change that is required from a client user perspective is that the user needs to specify as the remote address on the Telnet command the localhost address and the remote port 11100.

The scenario consists of the following subtasks:

1. Start an i5/OS PASE (Portable Solutions Application Environment) shell session on the client side (i5OSP4) and prepare the session for public key authentication using the **ssh-agent** and **ssh-add** programs.
2. Establish an SSH tunnel between the i5OSP4 and i5OSP2 systems.
3. Start a Telnet session from the i5OSP4 system to the i5OSP2 system using the SSH tunnel.

Perform the following steps to establish and use the SSH tunnel:

1. If you have not already done so, establish a 5250 session from your workstation to the i5OSP4 system and sign on, in our scenario, with user BARLEN4.
2. Enter the i5/OS PASE shell with the command:

```
CALL QP2TERM
```

3. Enter the following command to start the **ssh-agent** program:

```
ssh-agent $SHELL
```

**Tip:** For a successful start of the agent, you do not receive any messages.

4. Add your private key to the agent's memory by using the following command:

```
ssh-add
```

The **ssh-add** command tries to read all private key files from the user's **.ssh** directory. For passphrase protected key files, the **ssh-add** program asks you for the passphrase that you used to protect the private key file. In this case, we use the private key file that was created in "Setting up the public and private keys" on page 28.

```
$
> ssh-agent $SHELL
> ssh-add
Identity added: /home/BARLEN4/.ssh/id_rsa (/home/BARLEN4/.ssh/id_rsa)
```

Figure 5-2 Output of running the ssh-add command

5. Start the SSH tunnel by using the following command:

```
ssh -T -L 11100:localhost:23 barlen2@i5OSP2
```

This command starts an SSH tunnel from port 11100 on the localhost address. The tunnel ends on port 23 on the i5OSP2 system and is authenticated as user BARLEN2.

6. From another 5250 session to the i5OSP4 system, enter the following command:

```
netstat *cnn
```

Find the job that is listening on port 11100. This is the local tunnel end on the i5OSP4 system.

7. Test the tunnel. Start a Telnet session using the following command:

```
TELNET RMTSYS(LOCALHOST) PORT(11100)
```

Notice that the Signon display is the one for the remote system, even though you did a Telnet to the localhost. This is proof that the connection uses your SSH tunnel.

**Important:** When using the previous command from an i5/OS PASE shell session, the shell session with the SSH tunnel must be kept active all times to allow any user on system i5OSP4 to establish a Telnet session via the tunnel. Even though the previous approach does everything we want, it is cumbersome to have somebody start a tunnel after each initial program load (IPL) and then keep the session with the tunnel up all times. The following method allows you to automatically start SSH tunnels and keep them active at all times:

```
SBMJOB CMD(CALL PGM(QP2SHELL) PARM('/QOpenSys/usr/bin/ssh' '-T' '-N'
'-L 11100:localhost:23' 'i5OSP2')) JOB(SSHTUN23) JOBQ(SSHJOBQ)
```

In this case, the SSH tunnel session is established via a batch job. You can then submit this job in the system startup program or a subsystem autostart job. It is a more complex call than entering the **ssh** command in an interactive shell. You should also use a separate job queue as shown in the previous command to separate **ssh**-related work from other batch jobs. For this, use the environment as documented in 2.3, “Starting the sshd daemon in a dedicated subsystem environment” on page 9.

Each parameter to the **ssh** command must be separately specified in the program call. The **-N** parameter causes the program to not submit any command to the **sshd**. It only establishes a forwarding tunnel that runs in the background.

The user that submits the job or the user that is specified for the **USER** parameter of the Submit Job (SBMJOB) command must use public key authentication. Password authentication does not work, because in a batch job, there is nobody who can enter the password. The keys must be generated with an empty passphrase, so that the **ssh** client can automatically open the private key file without prompting for a passphrase. As with other key files too, the private key file for the tunnel user must also be protected. That means there should be no public authority to the key files at all.

## 5.2 Setting up an SSH tunnel between a workstation and i5/OS

An SSH tunnel can also be used from a PC workstation to i5/OS. In this section, you establish an SSH tunnel for port 23 (Telnet). Authentication is performed through public keys to avoid user ID and password prompting when the tunnel is established. PuTTY is used as the **ssh** client. At the end, you establish a 5250 session across the tunnel.

### Prerequisites

You must install a couple of utilities to use public key authentication with PuTTY. You can download these utilities from the Internet at:

<http://www.putty.nl/download.html>

The following utilities provide the equivalent support that you get in i5/OS with the **ssh**, **ssh-keygen**, **ssh-agent**, and **ssh-add** utilities:

- ▶ **PuTTY**: The equivalent of the **ssh** utility
- ▶ **PuTTYgen**: Corresponds to the **ssh-keygen** utility
- ▶ **Pageant**: Provides the functions of **ssh-agent** and **ssh-add**

**Important:** In the steps in the following sections, we presume that you have installed the previously mentioned utilities in the c:\ssh directory on your workstation.

### Setting up the tunnel with PuTTY

To set up the tunnel with PuTTY:

1. From your Windows workstation, start the PuTTY client.
2. Enter the following connection information:
  - Host Name (or IP address): i5OSP4
  - Port: 22
  - Protocol: **SSH**
3. In the Saved sessions parameter, enter a name for this connection (for example, SSH Telnet Tunnel), and click **Save**.
4. Under Category, select **Connection** → **SSH** → **Tunnels**.
5. In the Options controlling SSH tunneling panel (Figure 5-3), enter the following tunnel configuration information:
  - a. For Source port, type 11100.
  - b. For Destination, type i5OSP4:23.
  - c. Select **Local**.

This tunnel has the same characteristics than the one created in “Setting up an SSH tunnel between i5/OS environments” on page 34. The tunnel listens on your PC on port 11100, but ends on system i5OSP4 on port 23.

- d. Click **Add** to add the tunnel configuration to the profile.

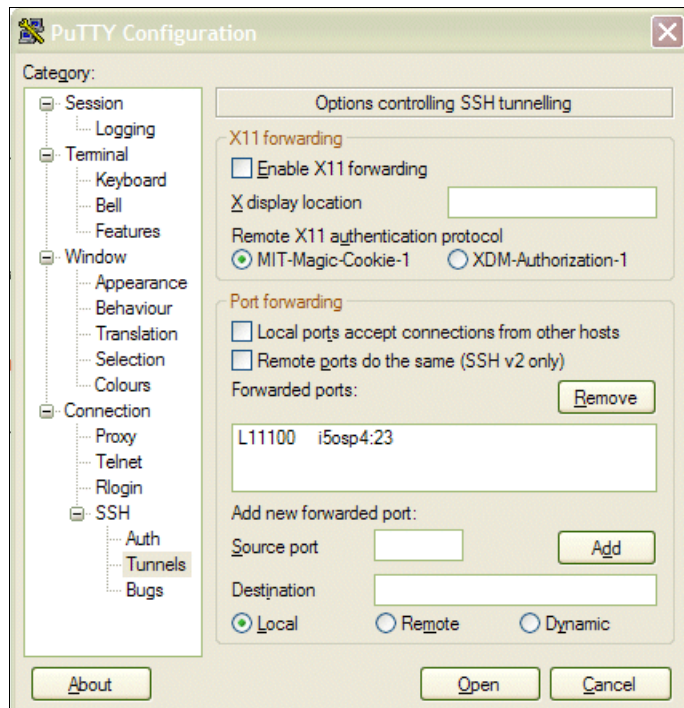


Figure 5-3 PuTTY tunnel settings

**Attention:** Use care with the PuTTY tunnel option *Local ports accept connections from other hosts*. This option enables your workstation to be a gateway for other computers. The local port, in our scenario port 11100, accepts Telnet connections from other systems and tunnels the connections to your tunnel. This can be a potential security risk. This option corresponds to the `ssh` option `-g`.

6. Additional ports must be tunneled when using the IBM iSeries Access for Windows PC5250 emulation. This emulation uses iSeries Access signon services, the port mapping service, and the remote command service when establishing a session.
  - a. Enter the following additional port to the configuration.
    - i. For Source port, type 449.
    - ii. For Destination, type i5OSP4:449.
  - b. Select **Local**.
  - c. Click **Add**.
  - d. Enter the following port:
    - i. For Source port, type 8470.
    - ii. For Destination, type i5OSP4:8470.
  - e. Select **Local**.
  - f. Click **Add**.
  - g. Enter the following port:
    - i. For Source port, type 8475.
    - ii. For Destination, type i5OSP4:8475.
  - h. Select **Local**.
  - i. Click **Add**.

- j. Enter the following port:
    - i. For Source port, type 8476.
    - ii. For Destination, type i5OSP4:8476.
  - k. Select **Local**.
  - l. Click **Add**.
7. Keep the PuTTY window open.

In the next steps, you prepare the client to perform public key authentication. You use the private key that you created in “Setting up the public and private keys” on page 28.

## Preparing the environment to use the keys

As previously mentioned, you use the private and public key pair that you already used with public key authentication between two i5/OS environments. To use these keys with PuTTY, you must prepare the environment as follows:

1. Establish a 5250 session from your Windows desktop to the i5OSP4 system and sign on. In this scenario we signed on with user BARLEN4. This was the user that already created a key pair in its .ssh directory.

2. Enter the i5/OS PASE shell with the following command:

```
CALL QP2TERM
```

3. Change to your user's .ssh directory:

```
cd /home/barlen4/.ssh
```

4. For the PuTTY client to authenticate under the BARLEN4 user profile with public key authentication, the public key must also be placed into the authorized\_keys file. (Remember that you performed this step on the i5OSP2 system when setting up public key authentication between i5OSP4 and i5OSP2, but not on the i5OSP4 system). To do this, enter the following command in your i5/OS PASE shell:

```
cat id_rsa.pub > authorized_keys
```

This command creates the authorized\_keys file and stores the public key in there.

5. Change the file and directory permissions as follows:

```
chmod go-w authorized_keys
chmod g-w /home/barlen4
chmod g-w /home/barlen4/.ssh
```

This command removes the w(rite) permissions from the primary group and from other users. The settings of the home and .ssh directory might have already shown the proper authority bit settings due to the steps performed in 3.1, “Preparing the user environment” on page 14.

6. Press F3 to exit the i5/OS PASE shell session.

You have completed the server side setup for public key authentication. In the remaining steps, you install the private key on your workstation. Remember that you use the same keys for authentication between i5/OS environments and from your workstation to i5/OS, because you are working under only one user. You can also generate another key pair on the client and then import its public key into the authorized\_keys file on the server. That way, you have to manage two different key pairs.

1. Open a command prompt on your Windows workstation.
2. Change to the ssh directory.

```
cd c:\ssh
```



3. Transfer your private key file from system i5OSP4 to the ssh directory using the FTP commands shown in Figure 5-4.

```
C:\ssh>ftp i5osp4
Connected to i5osp4.stgt.spc.ihost.com.
220-QTCP at i5osp4.stgt.spc.ihost.com.
220 Connection will close if idle more than 5 minutes.
User (i5osp4.stgt.spc.ihost.com:(none)): barlen4
331 Enter password.
Password:
230 BARLEN4 logged on.
ftp> ascii
200 Representation type is ASCII nonprint.
ftp> quote site nam 1
250 Now using naming format "1".
ftp> get /home/barlen4/.ssh/id_rsa id_rsa
200 PORT subcommand request successful.
150 Retrieving file /home/barlen4/.ssh/id_rsa
250 File transfer completed successfully.
ftp: 883 bytes received in 0.05Seconds 17.66Kbytes/sec.
ftp> quit
221 QUIT subcommand received.
```

*Figure 5-4 FTP session*

4. While in the ssh directory, enter the **puttygen** command to start the key management tool for PuTTY. This tool allows you generate key pairs and to import keys that were generated on another platform, such as the key that you just transferred to the workstation. The window changes as shown in Figure 5-5.

5. In the PuTTY Key Generator window (Figure 5-5), click **Load** to import your existing key and select the private key file c:\ssh\id\_rsa.

Keep in mind that you must select the **All Files (\*.\*)** file type to select the file. Click **Open** to select the file.



Figure 5-5 PuTTY Key Generator window

**Note:** Because the private key file was created without passphrase protection, the **puttygen** utility imported the key without prompting for a passphrase. If you created the private key file with passphrase protection, you see a prompt to enter the passphrase first, before **puttygen** can successfully import the key.

6. A confirmation message is displayed as shown in Figure 5-6. Click **OK** to close the message window.

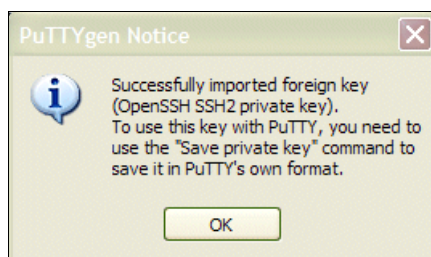


Figure 5-6 Key import confirmation message

7. The PuTTY Key Generator changes as shown in Figure 5-6. Click **Save private key** to save the key to the \ssh directory.

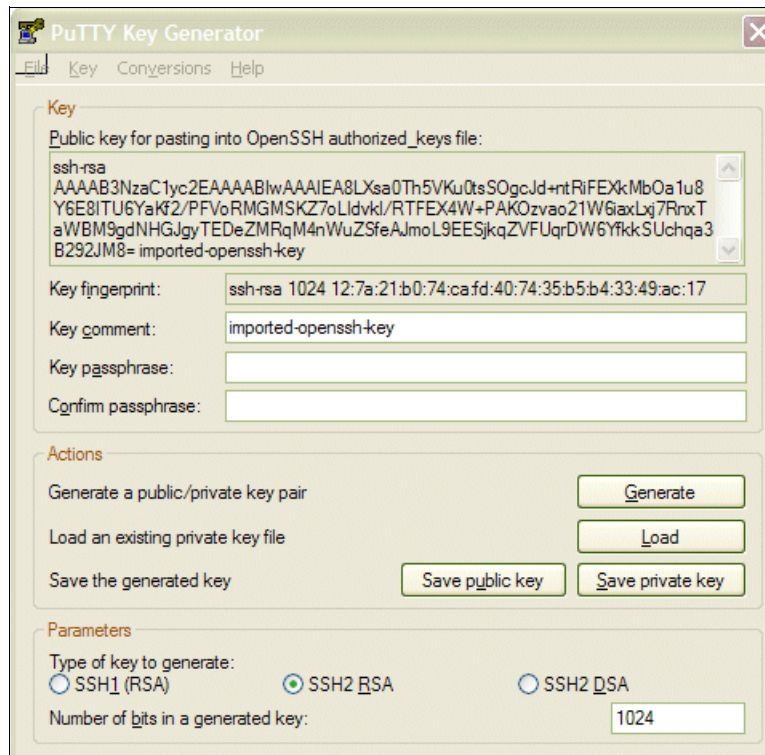


Figure 5-7 PuTTY Key Generator: Import complete

8. You must save the key into a new file, because PuTTY uses its own key file format. The utility warns you when you try to save the key without passphrase protection. Click **Yes** to the message to continue.
9. Enter the file name `bar1en4key.ppk` and make sure the store path is still `\ssh`.
10. Close the PuTTY Key Generator window.

## Defining the PuTTY public key authentication settings

Now modify the PuTTY settings to allow public key authentication.

1. Maximize the PuTTY window again and under Category, click **Connection** → **SSH** → **Auth**.
2. In the Options for controlling SSH configuration panel on the right (Figure 5-8), enter the path and file name of the private key file that you created in the Private key file for authentication field. In this example, we enter C:\ssh\barlen4key.ppk.

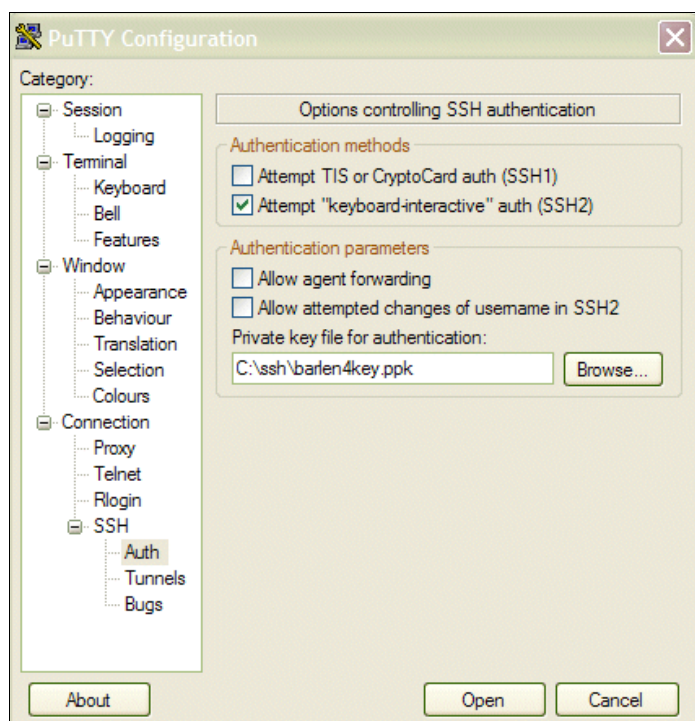


Figure 5-8 PuTTY authentication settings

3. Under Category, click **Session**.
4. To automate as much as possible, especially when establishing a tunnel, specify the user ID that belongs to the private key file. In the Basic options for your PuTTY session panel (Figure 5-9), enter the following information:
  - Host name (or IP address): barlen4@i5OSP4
  - Saved Sessions: SSH Tunnel to i5OSP4

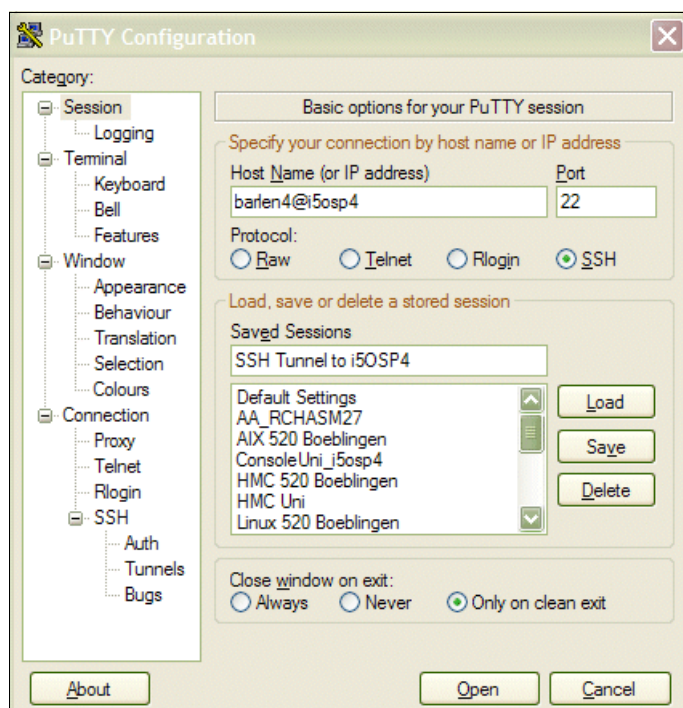


Figure 5-9 PuTTY session page

When using these settings, the PuTTY client tries to automatically authenticate the SSH session under the user name barlen4 on system i5OSP4.

5. Click **Save** to save the profile.

## Establishing the tunnel

In the PuTTY client, click **Open** to start the connection. As shown in Figure 5-10, you see that the client automatically tries to authenticate with barlen4. It also opens the private key file that you configured in the session profile. When passphrase protection is active, you are prompted to enter the passphrase.

```
Using username "barlen4".
Authenticating with public key "imported-openssh-key"
$
```

Figure 5-10 PuTTY login prompt

The tunnel is now established and can be used for Telnet traffic.

## Starting a secure 5250 Telnet session

It is time to test your secure tunnel.

1. Right-click the Windows desktop and select **New** → **iSeries Desktop Icon**.
2. Select **PC5250 Emulator** as the application and click **Next**.
3. For the iSeries system, enter `localhost` and click **Next**.
4. For the icon text, enter `Tunnel to i5OSP4` and then click **Next** and then click **Finish**.
5. On the Windows desktop, double-click the **Tunnel to i5OSP4** (🖨️) icon. The PC5250 properties page opens.
6. Verify that the following connection settings are specified:
  - a. For System name, select **localhost**.
  - b. For Port number, type 11100. The connection is established to the localhost (your PC workstation) to port 11100 (the begin of the tunnel).
  - c. Click **OK** to start the session.

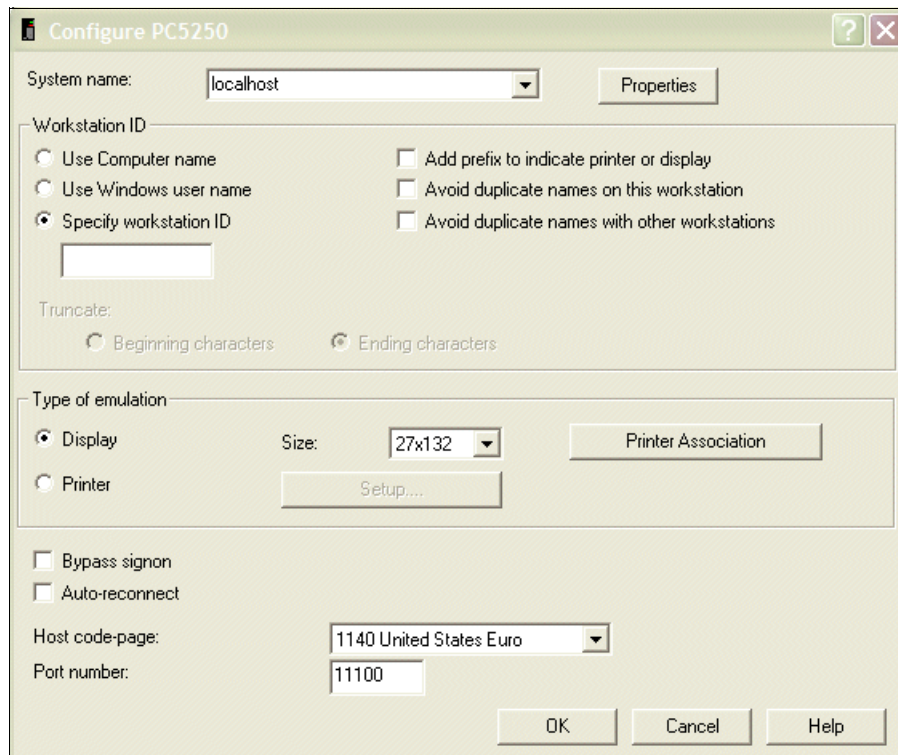


Figure 5-11 PC5250 connection settings

7. You can sign on to i5/OS and use the `NETSTAT *CNN` command to check the ports and addresses that are used for your connection. Notice that the remote IP address is the address of the iSeries system rather than the remote PC workstation.
8. Sign off and close your 5250 session.
9. In the open PuTTY window, enter `Exit` to end the session.



## 5.3 Automating the tunnel session start

As you have seen in the previous task, you had to open the PuTTY graphical interface, load the profile, and start it. When passphrase protection is active for the private key file, you are also prompted to enter the passphrase that protects the private key file. The next steps show you an alternative way to establish the tunnel through the command line interface.

Perform the following steps to meet the previously stated goals:

1. Using the Windows command prompt, change to the `\ssh` directory:

```
cd c:\ssh
```

2. From the command prompt in the `\ssh` directory, enter the following command to start the tunnel:

```
putty -load "SSH Tunnel to i50SP4"
```

This command loads the configuration profile that you created in the PuTTY graphical interface. It opens the `ssh` command prompt. When passphrase protection is defined for the private key file, you can use the **pageant** utility to preload private keys into memory before establishing the tunnel as described in , “Automatic setup of tunnels with private key file passphrase protection” on page 45.

```
Using username "barlen4".
Authenticating with public key "imported-openssh-key" from agent
$
```

Figure 5-12 SSH tunnel start messages

3. In the PuTTY command window, right-click the upper left corner of the window and select **Event Log** to display the ports that are tunneled with this connection.
4. Close the Event Log window.
5. In the PuTTY command prompt, enter the `exit` command to stop the tunnel and exit the SSH session.

### Automatic setup of tunnels with private key file passphrase protection

For better security, we recommend that you protect your private key file with a passphrase. This is especially recommended when using a Windows operating system and no proper resource protection for the private key file is defined. In this case, you can load the private key into memory before starting the tunnel by using the **pageant** utility. The following steps show you how to use the **pageant** tool to load a private key that is passphrase-protected.

1. Using the Windows command prompt, change to the `\ssh` directory:

```
cd c:\ssh
```

2. Enter the **pageant** command to start the PuTTY key agent. The program starts in the background and places an icon in the active program tasks bar.
3. Right-click the **pageant icon** (🔑) and select **Add Key** from the menu.
4. Select the `\ssh\barlen4key.ppk` private key file and click **Open**.

You can also load keys from the command line when starting **pageant**. To load your private key file with the command line options, you enter the following command string:

```
\ssh\pageant \ssh\barlen4key.ppk
```

Remember that the key file in this scenario is not passphrase protected.

5. When prompted to enter the passphrase, enter the passphrase that is used to protect the private key.
6. Double-click the **pageant** icon and verify that the key was added. See Figure 5-13.

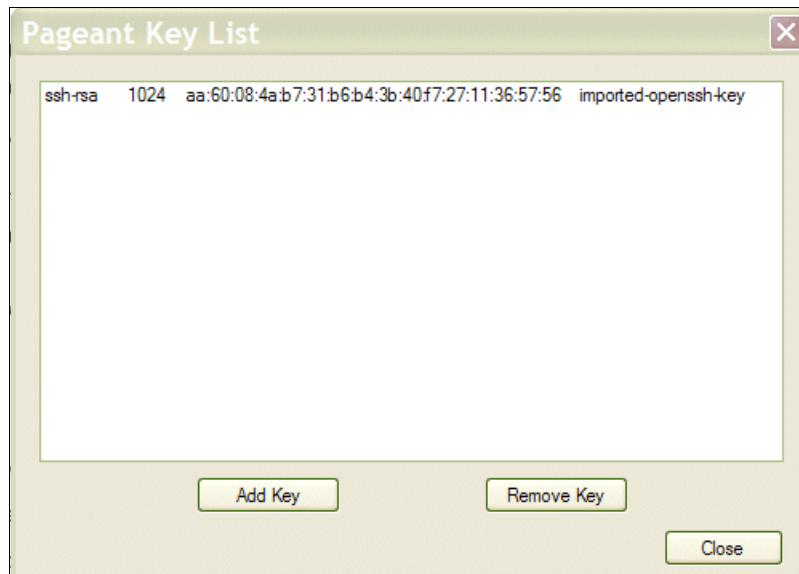


Figure 5-13 The pageant key list

7. From the command prompt in the \ssh directory, enter the following command to start the tunnel:

```
putty -load "SSH Tunnel to i5OSP4"
```

This command loads the configuration profile that you created in the PuTTY graphical interface. It opens the **ssh** command prompt. As you can see in Figure 5-14, the authentication takes place without entering a passphrase when PuTTY starts.

```
Using username "barlen4".
Authenticating with public key "imported-openssh-key" from agent
$
```

Figure 5-14 SSH tunnel start messages

8. In the PuTTY command prompt, enter the **exit** command to stop the tunnel and exit the SSH session.





## Using SSH to control your HMC

The Hardware Management Console (HMC) is a tool to manage logical partitions (LPAR) in an IBM System i5 environment. Traditionally, the HMC is operated through a graphical interface. The graphical interface provides the following options:

- ▶ Define and manage partitions
- ▶ Define and manage partition profiles
- ▶ Operate managed systems
- ▶ Perform dynamic LPAR operations, such as moving processor and memory resources

All these functions that are available through the graphical interface can also be performed through commands. These commands are entered via an SSH session. This support opens a wide range of possibilities for automating system tasks, such as initiating the move of memory from one partition to another partition through a command sent from i5/OS via `ssh` to the HMC. Another example is to move a tape controller from one partition to another partition to make the tape available to all partitions for backup operations.

This chapter covers the following topics:

- ▶ Setting up `ssh` on the HMC
- ▶ Setting up public key authentication so that HMC commands can be issued from i5/OS in unattended mode
- ▶ Moving resources between partitions using SSH in i5/OS

## 6.1 Setting up SSH on the HMC

To allow SSH connections and remote command execution on a HMC, you must perform initial setup steps on the HMC itself. These steps are also explained in the IBM Systems Hardware Information Center at the following Web address:

<http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphai/settingupsecurescriptexecutionsbetweensshclientsandthehmc.htm>

Perform the steps in the following sections to enable the HMC to allow remote command execution through SSH.

### Enabling SSH on the HMC

To enable SSH on the HMC:

1. At the HMC or through the WebSM interface, sign on to the HMC with a profile that has the authority to manage the HMC configuration.
2. In the Navigation area, expand **HMC Management** and click **HMC Configuration**.
3. In the Contents area on the right, click **Enable or Disable Remote Command Execution**.
4. When the Remote Execution Options window opens, select the **Enable remote command execution using the ssh facility** check box as shown in Figure 6-1.
5. Click **OK** to save the settings.

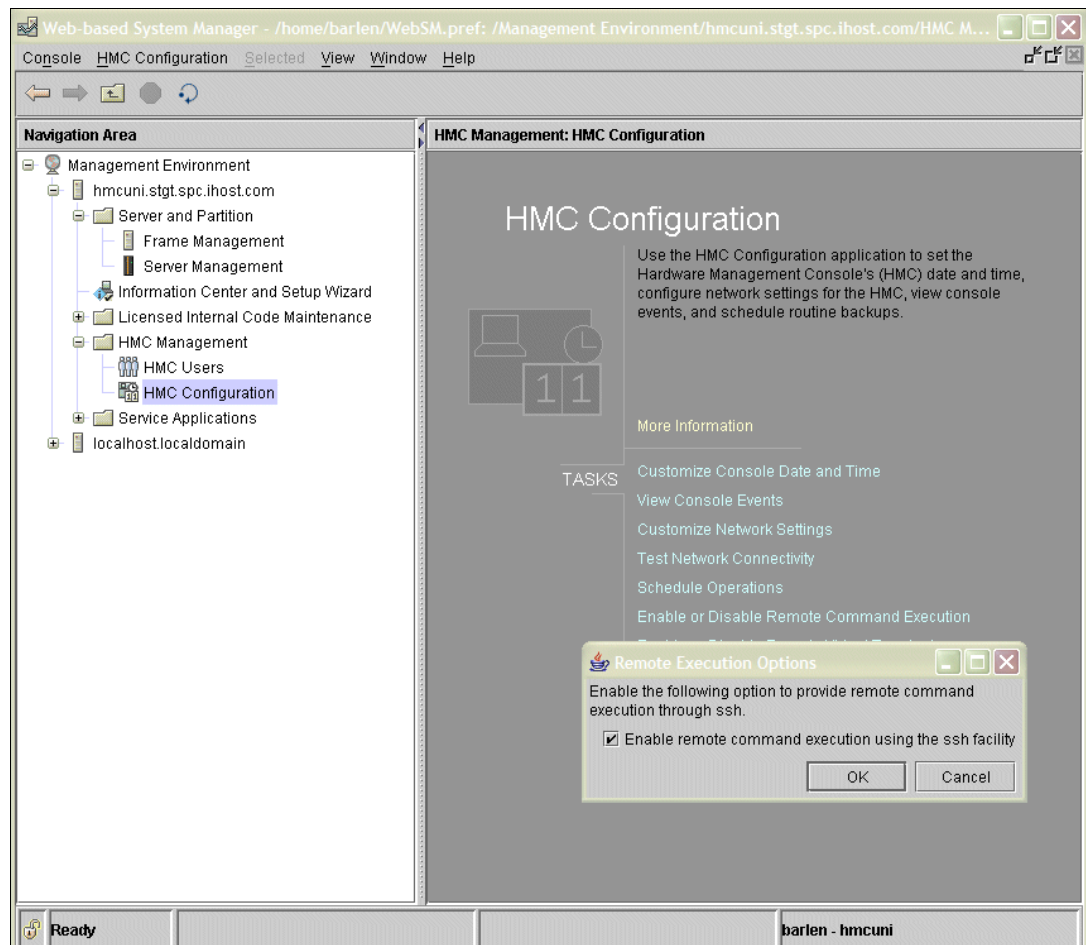


Figure 6-1 Enabling remote command execution on the HMC

## Creating a user account on the HMC

Create an HMC user with one of the following roles:

- ▶ Super administrator
- ▶ Service representative

Follow these steps:

1. In the navigation area, under HMC Management, click **HMC Users**.
2. In the HMC Users content pane, click **Manage HMC Users and Access**.
3. In the User Profiles window, select **User** → **Add** to add a new user account.

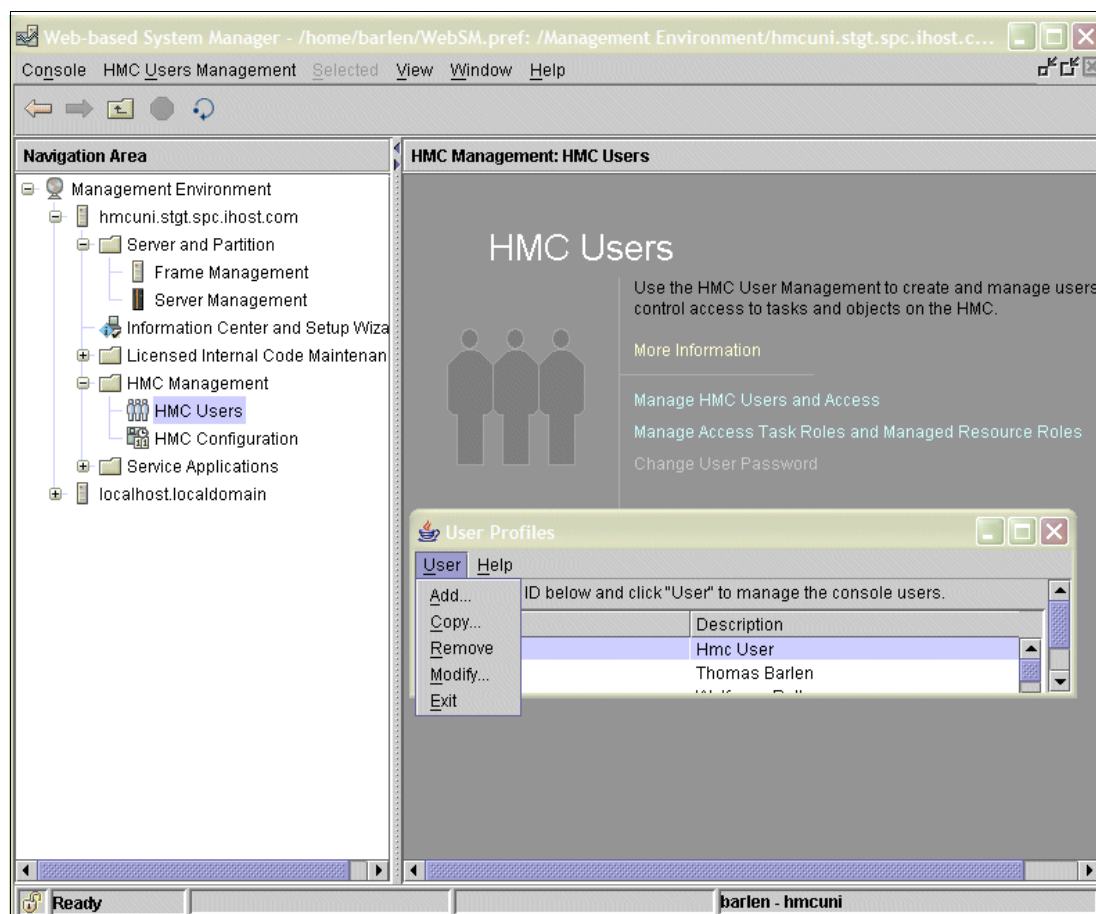


Figure 6-2 User Profiles window

4. In the Add User window (Figure 6-3), for the new user account, specify the information using the following parameters. The values provided for each option are ones that we used in this Redpaper. You might want to use different names.
  - a. For User ID, we type HMCSSH. Note that user ID names are case-sensitive.
  - b. For Description, we type HMC User for SSH.
  - c. For Password, enter a password that you will remember.
  - d. For Managed Resource Roles, we select **AllSystemResources**.
  - e. For Task Roles, we select **hmcsuperadmin**. You can also use service representative, but this user might not be able to perform all functions that you want to run via SSH.
  - f. Click **OK** to create the new user account.

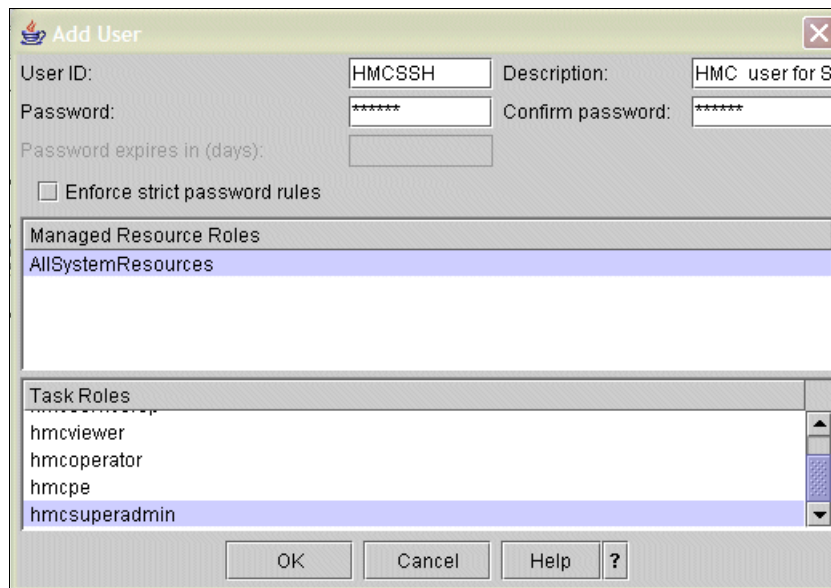


Figure 6-3 Add User window

5. In the User Profiles window, select **User** → **Exit** to close the window.

## 6.2 Setting up public key authentication

To submit an HMC command via SSH in unattended mode, you must set up public key authentication between the `ssh` client, in this case the i5/OS environment, and the HMC. Perform the following steps to set up public key authentication for the i5/OS user BARLEN4 on the i5OSP4 system to the HMC, with the host name HMCUNI and the user profile HMCSSH. After the setup is complete, user BARLEN4 on system i5OSP4 will be able to run HMC commands under user HMCSSH on the HMC.

1. User BARLEN4 on the i5OSP4 system needs a public key pair. Perform the steps described in “Setting up the public and private keys” on page 28 to create a public key pair for user BARLEN4 without private key passphrase protection.
2. Using an i5/OS PASE (Portable Solutions Application Environment) shell session on system i5OSP4, set the correct integrated file system permissions as follows:

```
chmod 700 /home/barlen4
chmod 700 /home/barlen4/.ssh
```

3. If you are not already in the .ssh directory, change to it via the shell command:

```
cd /home/barlen4/.ssh
```

4. Display the public key file by using the following command:

```
cat id_rsa.pub
```

The public key is displayed as shown in Figure 6-4.

```
$
> cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA8LXsa0Th5VKu0tsS0gcJd+ntRiFEXkMb0a1u8Y6E8ITU6YaKf2/P
FVoRMGMSKZ7oLIIdvk1/RTFEX4W+PAK0zvao21W6iaxLxj7RnxTaWBM9gdNHGJgyTEDeZMRqM4nWuZSfeAJmoL
9EESjkqZVFUqrDW6YfkkSUchqa3B292JM8= barlen4@i5osp4.stgt.spc.ihost.com
$
```

Figure 6-4 Public key

5. Add the public key to the authorized\_keys2 file on the HMC:

```
ssh HMCSSH@hmcuni "mkauthkeys --add
'ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA8LXsa0Th5VKu0tsS0gcJd+ntRiFEXkMb0
a1u8Y6E8ITU6YaKf2/PFVoRMGMSKZ7oLIIdvk1/RTFEX4W+PAK0zvao21W6iaxLxj7RnxTa
WBM9gdNHGJgyTEDeZMRqM4nWuZSfeAJmoL9EESjkqZVFUqrDW6YfkkSUchqa3B292J
M8= barlen4@i5osp4.stgt.spc.ihost.com'"
```

**Note:** You must enter the public key exactly as it was displayed with the `cat id_rsa.pub` command. This includes spaces and is in one string. The previous command establishes an SSH session to the HMC and signs on as user HMCSSH (case-sensitive). This was the user that was created previously in “Creating a user account on the HMC” on page 49. On the HMC, the `mkauthkeys` command is run with the `-add` parameter. This causes the key that is specified to be added to the `authorized_keys2` file on the HMC. Adding the key is required for public authentication.

Figure 6-5 shows the command processing output. Your output should look similar.

```
$
> ssh HMCSSH@hmcuni "mkauthkeys --add
'ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA8LXsa0Th5VKu0tsS0gcJd+ntRiFEXkMb0
a1u8Y6E8ITU6YaKf2/PFVoRMGMSKZ7oLIIdvk1/RTFEX4W+PAK0zvao21W6iaxLxj7RnxTa
WBM9gdNHGJgyTEDeZMRqM4nWuZSfeAJmoL9EESjkqZVFUqrDW6YfkkSUchqa3B292J
M8= barlen4@i5osp4.stgt.spc.ihost.com'"
The authenticity of host 'hmcuni (172.17.17.5)' can't be established.
. key fingerprint is RSA.
Are you sure you want to continue connecting (yes/no)?
> yes
Warning: Permanently added 'hmcuni,172.17.17.5' (RSA) to the list of known hosts.
HMCSSH@hmcuni's password: $
```

Figure 6-5 Command output for adding a public key

**Note:** When you establish an SSH session to the HMC the first time, you are prompted to accept the host key from the HMC. When you answer *Yes* to the question, the host key is added to the `known_hosts` file of user BARLEN4 on the i5OSP4 system.

6. When prompted, enter the password of user HMCSSH.

The public key authentication setup is now complete, and you can continue with running HMC commands.

## 6.3 Moving resources between partitions using SSH in i5/OS

Let us look closer at how you can use the `ssh` utility to move hardware resources in a dynamic logical partitioning (DLPAR) environment. The intent of this chapter is not to show you the HMC commands that are available and how to use all of them. Instead, it is to demonstrate how to use HMC commands from an i5/OS or any other environment that supports the SSH client. For more information about LPAR and HMC commands, refer to the IBM Redbook *Logical Partitions on System i5: A Guide to Planning and Configuring LPAR with HMC on System i*, SG24-8000.

The commands shown in this chapter are based on the LPAR environment shown in Figure 6-6.

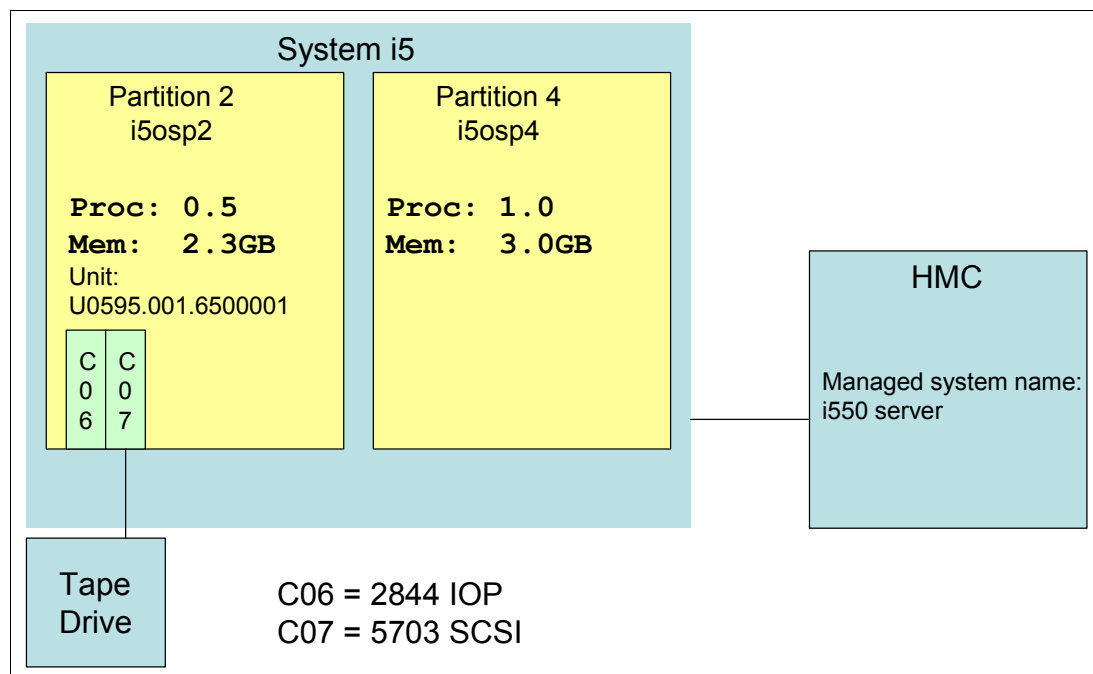


Figure 6-6 LPAR environment

### Moving processor units between partitions

In the first example, the 0.3 processing units are moved via DLPAR from partition 4 (i5OSP4) to partition 2 (i5OSP2). The move is initiated from i5/OS.

1. Sign on to system i5OSP4 with user BARLEN4. This is the user profile that is set up for public key authentication with the HMC.
2. Start an i5/OS PASE shell session by entering the CL command:

```
CALL QP2TERM
```

3. Move the processing units from partition 4 to partition 2:

```
ssh HMCSSH@hmcuni "chhwres -r proc -m 'i550 server' -o m -p I5OSP4 -t I5OSP2
--procunits 0.3"
```

You can use the **chhwres** command to add, move, or remove resources, such as input/output resources, virtual resources, processor, and memory. The meaning of the specified parameters is as follows:

- r** Specifies the type of resource to be processed. In this example, processor resources are moved.
- m** Defines the name of the managed system.

**Tip:** The HMC command to determine the name of the managed system or systems is:

```
lssyscfg -r sys -F name
```

- o** The action to be performed for the resource. In this case it is *m* (move).
- p** The name of the partition a resource is moved from.

**Tip:** You can use the following command to list all partition names for a managed system:

```
lssyscfg -r lpar -m 'i550 server' -F name
```

- t** The target partition name.
- procunits** The number of processing units to be processed.

The HMC commands are like any other kind of Linux or UNIX-type commands; they don't issue any success message when they complete successfully. Therefore, we recommend that you verify the result of the HMC command after its completion. For example, if you want to verify that the processing units were moved successfully to partition, you can use the following command:

```
ssh HMCSSH@hmcuni "lshwres -m 'i550 server' -r proc --level lpar --filter lpar_names=I50SP2 -F curr_proc_units"
```

Figure 6-7 shows the complete command output.

```
$
> ssh HMCSSH@hmcuni "lshwres -m 'i550 server' -r proc --level lpar --filter
lpar_names=I50SP2 -F curr_proc_units"
0.8
$
```

Figure 6-7 Command output

## Moving adapter resources

Submitting HMC commands from an i5/OS environment can also be used to move resources, such as a SCSI adapter from one partition to another partition. That way, you can have one tape drive installed on your system and use it from multiple partitions. A backup process can be organized as follows:

1. Partition 4 has a tape drive allocated and varied on. A backup batch job runs.
2. At the end of the backup job in partition A, the job varies off the tape drive and sends via the **ssh** utility a command to the HMC to move the tape adapter and controller to partition 2. When the move has completed successfully, the job starts a backup job in partition 2 by issuing the corresponding command through the Run Remote Command (RUNRMTCMD) CL command.

3. The backup job in partition 2 starts and varies on the tape drive. Afterwards it performs the backup operation. When finished, it varies off the tape drive and, via an `ssh` command, moves the controlling adapter back to partition 4.

You can imagine all different kinds of uses for moving resources based on events that are occurring in a partition. Based on the LPAR environment described in Figure 6-6 on page 52, the following example shows you how to move the adapter resource in slot C07 in unit U0595.001.6500001 from partition i5OSP4 to partition i5OSP2.

1. Sign on to partition i5OSP4 with user BARLEN4.
2. Start an i5/OS PASE shell session.
3. Enter the following command to move the adapter. Remember to vary off resources before trying to move resources through DLPAR.

```
ssh HMCSSH@hmcuni "chhwres -r io --subtype slot -m 'i550 server' -o m
-p I5OSP4 -t I5OSP2 -l 2102000A"
```

This command moves the resource in slot C07 from partition i5OSP4 to partition i5OSP2. As you can see, there is no parameter to specify a slot number. I/O slots are addressed by a DRC index number. Before you see how to determine the DRC index number, review the following explanation of the parameters:

- r** Specifies the type of resource to be processed. In this example, I/O resources are moved.
- subtype** When `io` is specified for the resource type parameter `-r`, you must provide more specific information about the type of I/O resource to be processed. In this case, the subtype is an adapter slot.
- m** Defines the name of the managed system.
- o** The action to be performed for the resource. In this case it is `m` (move).
- p** The name of the partition from which a resource is moved.
- t** The target partition name of the partition to which a resource is moved.
- l** Specifies the DRC index number. Each resource has a unique DRC index number. This is important, because you might have several I/O expansion units, such as a 0595, installed, and in each unit, you have an adapter in slot C07. One way to address the correct resource is by unit name and slot number, but the current implementation performs the addressing by a DRC index number.

## Determining the DRC index number

To find a DRC index number for a resource, you must know in which unit (system or expansion units) the resource is located. There are always several ways to reach the same result. The following method is one example to determine the DRC index number for the adapter in slot C07 in unit U595.001.6500001.

Enter the following HMC command to list all physical resources including the DRC index number in unit U595.001.6500001:

```
ssh HMCSSH@hmcuni "lshwres -r io --subtype slot -m 'i550 server'
-F phys_loc,drc_index --header --filter units=U0595.001.6500001"
```

This command displays only the physical location (`phys_loc`) and DRC index (`drc_index`) attributes as specified in the `-F` parameter. It lists all physical I/O resources of type `slot` that are installed in unit U0595.001.6500001 as defined in the filter parameter. The output also includes a header.



Figure 6-8 shows an example of the command output.

```
> sssh HMCSSH@hmcuni "lshwres -r io --rsubtype slot -m 'i550 server'
-F phys_loc,drc_index --header --filter units=U0595.001.6500001"
phys_loc,drc_index
C06,2101000A
C07,2102000A
C08,2103000A
C01,2101000B
C02,2102000B
C03,2103000B
C04,2104000B
$
```

*Figure 6-8 HMC command output*

In this case, the resource in slot C07 has the DRC index number of 2102000A.

**Tip:** If you do not know the slot number and unit in which an adapter is installed, you can always use the Work with Hardware Resources (WRKHDWRSC) CL command in the partition in which the adapter is installed and display the resource details.

In the previous example, the SCSI adapter with the attached tape drive was moved from partition i5OSP4 to partition i5OSP2. To use the tape drive in partition i5OSP2, you must also move the controlling input/output processor (IOP) in position C06.



# Abbreviations and acronyms

<b>AFS</b>	Andrew File System
<b>CCSID</b>	coded character set identifier
<b>DLPAR</b>	dynamic logical partitioning
<b>FTP</b>	File Transfer Protocol
<b>GPL</b>	General Purpose License
<b>HMC</b>	Hardware Management Console
<b>IBM</b>	International Business Machines Corporation
<b>IPL</b>	initial program load
<b>ITSO</b>	International Technical Support Organization
<b>LPAR</b>	logical partitioning
<b>LPO</b>	License Program Option
<b>i5/OS PASE</b>	i5/OS Portable Application Solution Environment
<b>SCP</b>	Secure Copy
<b>SFTP</b>	Secure File Transfer Protocol
<b>SSH</b>	Secure Shell
<b>SSHD</b>	Secure Shell Daemon
<b>TCP</b>	Transfer Control Protocol



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 60. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *IBM eServer iSeries Security Guide for IBM i5/OS Version 5 Release 3*, SG24-6668
- ▶ *Logical Partitions on System i5: A Guide to Planning and Configuring LPAR with HMC on System i*, SG24-8000
- ▶ *Hardware Management Console (HMC) Case Configuration Study for LPAR Management*, REDP-3999

## Online resources

These Web sites are also relevant as further information sources:

- ▶ Portable Utilities for i5/OS  
<http://www.ibm.com/servers/enable/site/porting/tools/openssh.html>
- ▶ OpenSSH project  
<http://www.openssh.org/>
- ▶ OpenSSL project  
<http://www.openssl.org/>
- ▶ zlib compression  
<http://www.zlib.net/>
- ▶ Tunneling Database Traffic with openSSH and Linux  
<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0312lurie/>

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## Symbols

.ssh directory 16

## Numerics

5250 Telnet session 44

## A

adapter resources 53  
AFS Ticket Passing 4  
authorities 24  
authority settings 14  
authorized\_keys file 22, 24, 38

## B

batch mode 35  
scp command 27

## C

chhwres command 53  
CL batch program 29

## D

data compression 3  
dedicated subsystem environment 9  
DRC index number 54  
dynamic logical partitioning (DLPAR) 52

## F

file transfer 21

## H

Hardware Management Console (HMC) 47  
moving resources 52  
public key authentication 50  
user account 49  
users and access 49  
home directory 14  
permissions 14

## I

i5/OS implementation 4  
i5/OS PASE shell script 30  
IBM Portable Utilities for i5/OS 4  
id\_rsa 23  
id\_rsa.pub 23, 51

## K

Kerberos 4  
key pair generation 22

## L

LPAR 47

## M

mkauthkeys command 51

## O

OpenSSH 4, 21  
file transfer and public key authentication 21  
for i5/OS 1  
tools and files 2  
OpenSSL 4

## P

pageant command 36, 45  
passphrase 23, 29, 40  
protection 45  
password authentication alternative 22  
permissions 38, 50  
for the home directory 14  
port forwarding 3  
Portable Utilities for i5/OS license program 5, 7  
private key 38  
private key file passphrase protection 45  
processor units 52  
public key 3, 20, 38  
addition to authorized\_keys file 24  
transfer 23  
public key authentication 21–22, 50  
exploiting with ssh 31  
scp to transfer files 26  
PuTTY 2, 36  
establish SSH connection to i5/OS 18  
public key authentication settings 42  
SSH tunnel 36  
puttygen command 36, 39

## R

Redbooks Web site 60  
Contact us viii

## S

scp command  
in batch mode 27  
to transfer files 26  
scp file copy program 3, 21  
secure 5250 Telnet session 44  
sftp 3, 21  
SSH  
connection to i5/OS established by PuTTY 18  
enablement on the HMC 48

- moving HMC resources between partitions 52
- setup on the HMC 48
- to control your HMC 47
- ssh
  - client utility 2
  - public key authentication 31
  - utility, remote use to run commands 17
- SSH session 13, 20
  - from other platforms to i5/OS 18
  - home directory 14
  - home directory permissions 14
  - ssh between i5/OS environments 15
  - user environment 14
- SSH tunnel 33
  - automatic setup with private key file passphrase protection 45
  - automating session start 45
  - setup between i5/OS environments 34
  - setup between workstation and i5/OS 36
  - setup with PuTTY 36
- ssh\_config 2, 8
- SSH1 1
- SSH2 1
- ssh-add program 3, 26, 34
- ssh-agent program 3, 26, 34
- sshd daemon 2, 7, 9
  - in a dedicated subsystem environment 9
  - setup 8
  - setup and running 7
  - submit job start 9
  - system configuration modification 8
- sshd\_config 2, 4
- ssh-keygen command 3, 28
- Submit Job (SBMJOB) command 9
- subsystem environment 9

## T

- T switch 15, 21
- tunnel 34
- tunneling 33

## U

- user account 49
- user environment 14
- user profile 14

## X

- X11 forwarding 3

## Z

- zlib 4







# Securing Communications with OpenSSH on IBM i5/OS



**Redpaper**

**Learn how to install, configure, and use SSH with i5/OS**

**Discover how to control a Hardware Management Console through SSH**

**Explore SSH tunnels to protect network traffic**

The OpenSSH open source product is widely used to securely access command line shells remotely. Secure Shell (SSH) tunneling or port forwarding capabilities allow users to establish a secure link for data traffic that otherwise flows in the clear over communication links. Additional utilities, such as **scp** and **sftp**, provide secure file transfer services. On the IBM System i platform, OpenSSH is shipped as part of the license program option IBM Portable Utilities for i5/OS and is available for systems that run V5R3 and later.

This IBM Redpaper introduces you to the OpenSSH implementation and the included utilities in IBM i5/OS. It teaches you how to use SSH in i5/OS as a server (daemon) and as a client. It explains how you can set up public key authentication for better protection during authentication.

In addition, this paper discusses the configuration for port forwarding from i5/OS as well as from Microsoft Windows. It also explains how you can use SSH to control a dynamic logical partitioning (DLPAR) environment. Plus it explores how SSH can be used from i5/OS to control Hardware Management Console (HMC) tasks, such as dynamic resource allocation.

The purpose of this Redpaper is to show you how to set up and use OpenSSH in an i5/OS environment, not to cover all capabilities provided with OpenSSH. For a complete list of functions and online help, visit the OpenSSH Web site at the following address:  
<http://www.openssh.org>

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

## BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)