CygSSH User Guide

CygSSH is a convenient packaging of the Cygwin version of OpenSSH that provides a simple way to install and use the Cygwin version of OpenSSH on Windows.

This software is covered by the license agreements listed in appendix F.

Contents

Chapter 1: Installing the Package

The system requirements are the following:

- Windows 7 SP1 or or Windows Server 2008 R2 SP1 or later

Vista/Server 2008 support has been dropped to increase security of the installer.

The installer installs the appropriate version of the package based on the OS architecture; that is, it installs the 64-bit version on x64 versions of Windows, or the 32-bit version otherwise. (See section 8.1 for an important caveat about running the 32-bit version of the OpenSSH server service on 64-bit non-x64 downlevel OS versions.)

The installer performs the following tasks:

- Creates the needed directories and copies the files
- Configures file and directory access control lists (ACLs)
- Configures the `fstab` file to disable Cygwin file system permission changes (see section 7.1)
- If the setup type is **full** (see section 1.4), creates the OpenSSH server host keys (see section 6.1)
- If the setup type is **full** (see section 1.4), creates the local `sshd` account (see appendix A)
- If the setup type is **full** (see section 1.4), creates the local access group and updates the `sshd_config` file (see section 5.1)
- If the setup type is **full** (see section 1.4), adds a rule to the Windows firewall to allow communication to the OpenSSH server service
- If the setup type is **full** (see section 1.4) and the **startservice** task is selected (see section 1.6), starts the OpenSSH server service
- Performs any other tasks selected on the **Select Additional Tasks** page (see section 1.6)

The installer package is built using the Inno Setup installer, so it supports Inno Setup's command line parameters.

Example installation command:

`CygSSH-Setup-`*`version`*` /SILENT /CLOSEAPPLICATIONS`

Replace *version* with the version (e.g., **8.4.1.6**).

The above command installs the package silently (i.e., without user interaction). The `/CLOSEAPPLICATIONS` parameter automatically stops running services and closes programs so that the installer can reinstall or upgrade the files. Stopped services will be automatically restarted after the installation completes.

If you want to prevent the installer from displaying its progress window during a silent install, use `/VERYSILENT`

instead of /SILENT.

The installer also supports non-administrative install mode, which installs only the client files for the current user (does not require administrator privileges). To run in non-administrative install mode, use the /CURRENTUSER parameter on the command line.

The following sections describe the pages of the installer wizard.

1.1 License Agreement

The **License Agreement** page of the installer wizard lists the license agreements used by the various software components. You must accept the license agreements to continue with the installation.

If you run a silent install using the /SILENT or /VERYSILENT command line parameters, you are providing an implicit acceptance of all license agreements (see appendix F).

1.2 Information

The **Information** page of the installer wizard lists release notes for the current version of the package.

1.3 Select Destination Location

The **Select Destination Location** page of the installer wizard allows you to specify the directory path where the package will be installed. This is C:\Program Files\CygSSH by default (if C: is the system drive). If you started the installer using the /CURRENTUSER parameter, the default path will be like C:\Users*username*\AppData\Local\Programs\CygSSH (i.e., install only for the current user).

You can specify a different destination path on the installer's command line by using the /DIR="*installpath*" command line parameter (where *installpath* is the directory path name).

Throughout this documentation, the installation directory is referred to as *installpath*.

Note that this page doesn't appear if you are reinstalling or upgrading.

1.4 Select Setup Type

The **Select Setup Type** page of the installer wizard allows you to choose the setup type. The setup types are as follows:

- **full** - Full installation (server and client files)
- **client** - Client installation (client files only)

You can specify the setup type on the command line using the /TYPE="*type*" command line parameter (where *type* is either **full** or **client**). The default setup type is **full**, so the only reason to use the /TYPE command line parameter is if you want to specify /TYPE="client" as the default (typically for a silent install).

Note that this page doesn't appear under the following conditions:

- If you specified /CURRENTUSER on the installer's command line, because /CURRENTUSER implies the **client** setup type
- If you previously installed using the **full** setup type, because selecting the **client** setup type does not remove the server files

1.5 Select Start Menu Folder

The **Select Start Menu Folder** page of the installer wizard allows you to specify the name of the Start menu folder where it will install application shortcut icons.

You can specify the folder name on the command line using the `/GROUP="`*name*`"` command line parameter (where *name* is the folder name). Alternatively, you can specify `/NOICONS` to prevent creation of any Start menu shortcuts.

Note that this page doesn't appear if you are reinstalling or upgrading.

1.6 Select Additional Tasks

The **Select Additional Tasks** page of the installer wizard allows you to select additional tasks the installer should perform. The additional tasks are as follows:

- **startservice** - Corresponds to the **Start OpenSSH server service** check box. Starts the OpenSSH server service. This task is not applicable if the setup type is **client** (see section 1.4). This task is selected by default.
- **modifypath** - Corresponds to the **Add to system Path** (or **Add to user Path**) check box. Adds the *installpath*`\bin` directory to the system or user Path environment variable if it does not already exist. This task is selected by default. (If you started the installer with the `/CURRENTUSER` parameter, this checkbox adds to the user Path rather than the system Path.) (Note that this task does not appear if *installpath*`\bin` is already in the system or user Path.)
- **s4ulogonfix** - Corresponds to the **Implement MsV1_0S4ULogon fix** check box (see section 8.2). This task only appears in downlevel OS versions and is not applicable if the setup type is **client** (see section 1.4). This task is selected by default *only* under the following conditions:
    - The current operating system is a downlevel version (in fact, this task doesn't appear at all if you are not running a downlevel OS version)
    - You are installing interactively (i.e., you have not specified the `/SILENT` or `/VERYSILENT` command line parameters)
    - The current computer is *not* a domain member

You can specify one or more of the additional tasks on the command line using the `/TASKS="`*task*[`,`*task*[...]]`"` command line parameter (where *task* is one or more of the above task names, separated by `,` characters).

For an interactive (i.e., not silent) installation, the `/TASKS` parameter specifies which tasks are selected by default; you can change the defaults on the page as you install.

For a silent installation, the `/TASKS` parameter dictates the list of tasks the installer will perform. For example, if you specify `/TASKS="startservice" /SILENT` on the installer's command line, the installer will **only** perform the `startservice` task.

To indicate ‘no tasks’, specify `/TASKS=""` on the installer's command line.

Note that if you do not select the `modifypath` task when you install, you will need to do one of the following before running commands:

- Change to the *installpath*`\bin` directory before running the command
- Specify the path to the command when you want to run it

Chapter 2: Reinstalling or Upgrading the Package

Reinstalling or upgrading the package is as simple as running the installer:

- If the installer is the same version as the installed version, it will reinstall the installed version.
- If the installer is a newer version than the installed version, it will upgrade the installed version.

A reinstall or upgrade cannot proceed if the OpenSSH server service or any of the programs in the package are currently running. If any are running, the installer will prompt for permission to stop running services and close any programs.

You can prevent the prompt from appearing by specifying the `/CLOSEAPPLICATIONS` parameter on the installer's command line. This is particularly useful when using the `/SILENT` parameter (see chapter 1).

Chapter 3: Uninstalling the Package

To uninstall the package, use the standard means of accessing installed applications on your version of Windows (e.g., the **Programs and Features** applet in the Control Panel), and uninstall it from there.

If you want to uninstall silently (without user interaction), do the following:

1. Determine the filename of the uninstall program. The file is in *installpath* and is named like `unins`*nnn*`.exe` (where *nnn* is 3 digits - normally `000`).
2. Run the executable with the `/SILENT` command line parameter.

For example, if the uninstaller executable is `C:\Program Files\CygSSH\unins000.exe`, the command line to uninstall silently is the following:

```
"C:\Program Files\CygSSH\unins000.exe" /SILENT
```

You can also replace `/SILENT` with `/VERYSILENT` to prevent the uninstaller from displaying its progress window.

The uninstall program does not remove the following items:

- Configuration files in the `etc` directory
- The `lastlog` file in the `var\log` directory
- Any other user created files
- The local `sshd` user account

In addition, if you are running a downlevel OS (see chapter 8), the unistall program does not reverse the changes made by the `s4ulogonfix` installer task (see section 1.6). If you want to reverse this change, run the following command from an elevated Windows PowerShell window before uninstalling:

```
Set-S4ULogonFix -Disable
```

(See section 8.2 for details about what this command does.)

Chapter 4: Client Authentication

This chapter describes how to authenticate to an OpenSSH server using SSH client software.

4.1 Using the SSH Client Commands

The CygSSH package provides several SSH client commands:

- The **scp** command copies files to an SSH server
- The **ssh** command establishes an interactive shell session with an SSH server
- The **sftp** command establishes a file transfer session with an SSH server

Note: It's recommended to run the **ssh** and **sftp** SSH client commands using the **mintty** command (e.g., **mintty ssh** rather than just **ssh**). If you run these SSH client commands without the **mintty** command, line wrapping and other features may not work as expected.

Example 1:

```
mintty sftp COMPUTER1+localuser@computer1
```

This command establishes a file transfer session to `computer1` using the account `COMPUTER1+localuser`.

Example 2:

```
mintty ssh -i /cygdrive/c/Users/KenDyer/.ssh/kenprivkey KenDyer@computer2
```

This command establishes an interactive shell session to `computer2` using the `KenDyer` account using public key authentication (see section 6.2), using the file `C:\Users\KenDyer\.ssh\kenprivkey` as the private key.

Example 3:

```
scp "/cygdrive/C/Program\ Files/CygSSH/etc/sshd_config"
  "bkpsvc@server1:/cygdrive/C/Program\ Files\CygSSH\etc"
```

(All on one line) This command copies the `sshd_config` file from the current computer to a remote computer.

When using the Cygwin SSH client commands, note that path names must be in POSIX format, using the `\` character to ‘escape’ spaces where necessary (see section 7.3).

If you haven't connected to an SSH server before, the command will prompt you to store the server's host key (see section 6.1) in the `known_hosts` file. The default path of the `known_hosts` file is in the `.ssh` directory in the user's home directory (see section 7.4).

You can also use the **rsync** command over an SSH connection to transfer files more efficiently (see appendix B).

See appendix C for links to the documentation for the **ssh**, **scp**, **sftp**, and **rsync** commands.

4.2 Using Other SSH Clients

See the documentation for the SSH client for more information on how to use it to connect to an SSH server.

Popular Windows SSH clients include the following:

PuTTY - https://www.chiark.greenend.org.uk/~sgtatham/

WinSCP - https://winscp.net/

Chapter 5: Server Authentication

This chapter describes how to configure the OpenSSH server service to permit logons using password authentication (see section 5.2) or public key authentication (see section 5.3).

5.1 Using the Local Access Group

By default, the OpenSSH server service only allows accounts that are members of the `SSH Users` local group to log on. The installer (see chapter 1) will create the `SSH Users` local group if it does not exist. This local group is referred to as the *local access group*.

The local access group is empty by default, which means you must explicitly add accounts to the local access group to allow logon via SSH.

If the computer is an Active Directory domain member, nested domain groups are supported. For example, you can add a domain group as a member of the `SSH Users` local group, and all members of the domain group will be granted access to log on using SSH.

If you have an Active Directory domain, you can manage the membership of the `SSH Users` local group using a Group Policy Object (GPO).

The installer (see chapter 1) automatically updates the `sshd_config` file at installation time with the following line:

```
AllowGroups "[computername+]SSH Users"
```

If the computer is a domain member, the installer adds the computer name and + prefix (see section 7.2). Otherwise, the computer name and + prefix are not used.

If the computer name changes, or if the computer is removed as a domain member, you will need to manually edit the `sshd_config` file as follows:

- If the computer name changes, update the computer name on the `AllowGroups` line of the file.
- If the computer is removed as a domain member, remove the computer name and + character on the `AllowGroups` line.

Once complete, save the `sshd_config` file and restart the OpenSSH server service.

5.2 Using Password Authentication

Once an account is a member of the local access group on the OpenSSH server (see section 5.1), that account can log on using an SSH client using the account's password. No further configuration is required.

The account name must use Cygwin format (see section 7.2).

5.3 Using Public Key Authentication

Public key authentication is a means of authenticating an SSH logon without needing the account's password (see section 6.2).

To authenticate an account using a public key instead of a password, you must first create a key pair (see section 6.2.1).

Once you have a key pair, you will need to complete all of the following configuration steps to enable public key authentication for the account:

- Ensure the account is a member of the local access group (see section 5.1)
- Create the directories on the server (see section 5.3.1)
- Grant the account permission to the directories (see section 5.3.2)
- Add the account's public key to the `authorized_keys` file (see section 5.3.3)

5.3.1 Create the Directories on the Server

To allow an account to authenticate using a public key (see section 5.3), you must create a directory named for the account in the *installpath*`\users` directory, and a `.ssh` directory within that directory.

This section also applies to setting up an SFTP-only account (see section 5.4).

The account name must use Cygwin format (see section 7.2).

For example, suppose the current computer is a domain member, and you want to allow the `KenDyer` account in the computer's domain to authenticate.

In this example you would create the following directories on the server:

- *installpath*`\users\KenDyer`
- *installpath*`\users\KenDyer\.ssh`

5.3.2 Grant the Account Permission to the Directories

To allow an account to authenticate using a public key (see section 5.3), you must update the directory's access control list (ACL) to grant the account at least read (**RX**) access to the directory and it subdirectories.

This section also applies to setting up an SFTP-only account (see section 5.4).

You can use the **setacl** command to grant the permissions.

First, open an elevated PowerShell console (i.e., right-click the PowerShell icon and choose `Run as administrator`).

Run a command like the following:

```
setacl -on "installpath\users\cygwinaccountname" -ot file
  -actn ace -ace "n:windowsaccountname;p:read"
```

(All on one line) Replace *cygwinaccountname* with the Cygwin account name (see section 7.2), and replace *windowsaccountname* with the Windows account name.

For example, suppose the current computer is a member of the `FABRIKAM` domain and you are setting the permissions for the `KenDyer` account in that domain. The command would be as follows:

```
setacl -on "C:\Program Files\CygSSH\users\KenDyer" -ot file
  -actn ace -ace "n:FABRIKAM\KenDyer;p:read"
```

(All on one line)

If you want to grant change (**RWXD**) instead of just read (**RX**) access, replace `read` at the end of the `-ace` parameter's argument with `change` (i.e., `p:change` instead of `p:read`).

5.3.3 Add the Account's Public Key to the `authorized_keys` File

To allow an account to authenticate using a public key (see section 5.3), you must add the account's public key to the `authorized_keys` file.

The `authorized_keys` file is a text file in the account's `.ssh` directory that contains one line for each public key for the account. (The OpenSSH documentation describes the `authorized_keys` file format.)

If you use the **New-SSHKey.ps1** script to generate a key pair (see section 6.2.1.1), you can use the generated public key (`.pub`) file as the `authorized_keys` file for the account.

If you use **PuTTYgen** to generate a key pair (see section 6.2.1.2), you can use the text box labeled **Public key for pasting into OpenSSH authorized_keys file** as the content of the `authorized_keys` file. If you have a `.ppk` file, open it in **PuTTYgen**. It will put the public key in the text box, and you can copy it from there (make sure there are no line breaks).

For example, suppose the current computer is a member of the `FABRIKAM` domain, and you want to allow the `KenDyer` account in that domain to authenticate.

In this example, the file *installpath*`\users\KenDyer\.ssh\authorized_keys` would contain the public key(s) used to authenticate the `KenDyer` account.

5.4 Configuring an SFTP-Only Account

This section describes how to configure an SFTP-only account (i.e., the account can only connect to the OpenSSH server using a file transfer client such as **sftp** or **WinSCP**).

To configure an SFTP-only account, first create the directories for it (see section 5.3.1) and set the permissions (see section 5.3.2).

Next, create an `sftp` directory inside the account's directory.

Next, add lines like the following to the end of the `sshd_config` file:

```
Match User accountname
        X11Forwarding no
        AllowTcpForwarding no
        PermitTTY no
        ChrootDirectory /users/%u/sftp
        ForceCommand internal-sftp
```

Replace *accountname* with the account's Cygwin account name (see section 7.2).

The `ChrootDirectory` setting is not a true security control on Windows and is used only to limit the file system view on SFTP-only accounts (see appendix A).

Once complete, save the `sshd_config` file and restart the OpenSSH server service.

Note: You can't prevent the `cygdrive` and `dev` directories from appearing on the client side of the SFTP connection, but they don't cause any problems, either.

Example configuration:

Suppose the current computer is a domain member named `COMPUTER1`, and local account `sftponly` will be an SFTP-only account.

In this example, first create the following directories on the OpenSSH server:

- *installpath*`\users\COMPUTER1+sftponly`
- *installpath*`\users\COMPUTER1+sftponly\sftp`

Next, grant the `sftponly` account at least read (**RX**) access to the *installpath*`\users\COMPUTER1+sftponly` directory and its subdirectories.

Then, add the following lines to the end of the `sshd_config` file:

```
Match User COMPUTER1+sftponly
        X11Forwarding no
        AllowTcpForwarding no
        PermitTTY no
        ChrootDirectory /users/%u/sftp
        ForceCommand internal-sftp
```

If the account will authenticate using a public key instead of a password (see section 5.3), add the `AuthenticationMethods publickey` setting; e.g.:

```
Match User COMPUTER1+sftponly
        AuthenticationMethods publickey
        X11Forwarding no
        AllowTcpForwarding no
        PermitTTY no
        ChrootDirectory /users/%u/sftp
        ForceCommand internal-sftp
```

(You must also add the `authorized_keys` file to the account's `.ssh` directory, as described in section 5.3.3.)

Once complete, save the `sshd_config` file and restart the OpenSSH server service.

Chapter 6: Public Key Cryptography

Public key cryptography, sometimes also called *asymmetric cryptography*, is an encryption technique that uses two cryptographically related signatures--a *public key* and a *private key*. (The public key and private key are typically referred to as a *key pair*.) The public key is used to encrypt, and the private key is used to decrypt. It is not computationally feasible to generate a private key based on a public key.

OpenSSH uses public key cryptography to ensure the communication between an SSH client and the OpenSSH server are encrypted.

6.1 Host Keys

A *host key* is a cryptographic signature that uniquely identifies an OpenSSH server to SSH clients.

The installer (see chapter 1) automatically creates the necessary host key files at installation time.

The files containing the host keys are stored in *installpath*`\etc` and are named using the format `ssh_host_`*keytype*`_key[.pub]`, where *keytype* is **dsa**, **ecdsa**, **ed25519**, or **rsa**. The `.pub` suffix indicates the public key. Each key type represents an encryption algorithm supported by OpenSSH.

For example, the file `ssh_host_ed25519_key` contains the private host key for the ED25519 encryption algorithm, and the file `ssh_host_ed25519_key.pub` contains its associated public key.

The private key files are configured with restricted access control lists (ACLs) to prevent unauthorized access (**SYSTEM:F** and **Administrators:F** only).

SSH clients can store an SSH server's host key for later comparison. If the host key is different the next time the client connects, it may indicate the server has been compromised. Host keys are a mechanism to validate the identity of the server to which you're connecting.

The Cygwin SSH client commands (see section 4.1) store known host keys in the `known_hosts` file in the current user's home directory (see section 7.4).

6.2 Public Key Authentication

Public key authentication is a means of authenticating an SSH logon without needing the account's password.

To use public key authentication to authenticate an account in OpenSSH, you will need to create a key pair. The private key is used on the client side of the SSH connection, and the public key is used on the server side.

The account name must use Cygwin format (see section 7.2).

6.2.1 Creating a Key Pair

This section describes different ways to create a key pair.

6.2.1.1 Creating a Key Pair Using the `New-SSHKey.ps1` Script

The **New-SSHKey.ps1** script is a wrapper for the **ssh-keygen** command. Using the script is straightforward: Open a PowerShell command window, run the script, it will prompt for the information it needs:

1. It will prompt you to enter the name of the private key file. The default path is the `.ssh` directory in the current user's home directory (see section 7.4). Press `Enter` without entering anything to use the default file name or enter a new file name.

2. It will prompt you to enter a passphrase. If you specify a passphrase, you must enter it every time you want to use the key.

3. It will prompt you to enter a comment. The comment is intended to help you identify the key. Press `Enter` without entering anything to use the default comment, or enter a new comment.

Sample command output follows:

```
PS C:\Users\accountname> New-SSHKey
Key type: ed25519
Enter file in which to save the key (C:\Users\accountname\.ssh\id_ed25519):
Enter passphrase (leave empty for no passphrase):
Confirm passphrase (leave empty for no passphrase):
Comment (accountname@hostname): My SSH key
Generating public/private ed25519 key pair.
Your identification has been saved in /cygdrive/c/Users/accountname/.ssh/id_ed25519.
Your public key has been saved in /cygdrive/c/Users/accountname/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:w5UB4i5qymXdQX/7ErHgbglow77s6Q6h/kaN0iavKbA My SSH key
The key's randomart image is:
+--[ED25519 256]--+
|  E...+o+.=o.    |
| . oo o+ =.=o    |
|. =.o*  . ...    |
| o Bo .. =       |
|  = + . S        |
| = . = o o       |
|o o   = .        |
|.  . +.+         |
|    o.o..        |
+----[SHA256]-----+
Restricted permissions on file 'C:\Users\accountname\.ssh\id_ed25519'.
```

The **New-SSHKey.ps1** script will attempt to restrict the permissions on the private key file to protect it from unauthorized access, but it may not be able to do this if the private key file is saved in a location where you don't have sufficient permissions to change the file's access control list (ACL). It's strongly recommended to store the private key in a location where you have permission to change the file's ACL so it will be protected from unauthorized access.

The file paths output by the **ssh-keygen** command are in POSIX format (see section 7.3).

6.2.1.2 Create a Key Pair Using PuTTYgen

To create a key pair using **PuTTYgen**, do the following:

1. Open the **PuTTYgen** program.

2. At the bottom of the **PuTTYgen** window, select **ED25519** as the type of key to generate.

3. Click the `Generate` button.

4. Move the mouse in the **PuTTYgen** window to generate randomness.

5. The text box near the top of the **PuTTYgen** window (labeled **Public key for pasting into OpenSSH authorized_keys file**) contains the public key for use on the OpenSSH server. (The line of text in this text box corresponds to the `.pub` file generated using the **New-SSHKey.ps1** script.)

6. You can specify a comment and/or a passphrase in the corresponding text boxes.

7. Click the `Save` button to save the private key in PuTTY (`.ppk`) format.

It's strongly recommended to store the private key file in a location with a restricted access control list (ACL), particularly if the private key is not protected with a passphrase.

6.2.2 Updating a Private Key

You can use the **Edit-SSHKey.ps1** script to update the passphrase or comment of a private key file. The command syntax is as follows:

```
Edit-SSHKey "filename" -Passphrase
```

or:

```
Edit-SSHKey "filename" -Comment
```

Replace *filename* with the name of the private key file.

The script will prompt you to enter a new passphrase or comment.

If you specify `-Comment`, the script will create a replacement public key file with a new comment (same filename but with `.pub` appended).

Hint: If you lose the public key file associated with a private key file, simply run **Edit-SSHKey.ps1** with the `-Comment` parameter. Enter an updated comment, and a new public key file will be created (same filename as private key, but with `.pub` appended).

Chapter 7: About Cygwin

Cygwin is a set of Windows DLLs (dynamically linked libraries) that provides substantial POSIX-compatible API (application programming interface) compatibility.

In effect, Cygwin is a POSIX emulation layer that allows programmers to compile POSIX-compatible source code into Windows executables with minimal changes.

This has several effects on programs that use the Cygwin emulation layer, as described in the following sections.

7.1 Cygwin File System Permissions

By default, Cygwin programs will attempt to reproduce POSIX-style file permissions using Windows access control lists (ACLs). This can result in unexpected permission problems when running Cygwin-based file management tools such as **rsync**.

To avoid this problem, the installer (see chapter 1) configures the file systems in the *installpath*`\fstab` file to use the `noacl` option for all directories.

Please see the following references in the Cygwin documentation for more information:

- File Permissions - `https://cygwin.com/cygwin-ug-net/ntsec.html#ntsec-files`
- The Cygwin Mount Table - `https://cygwin.com/cygwin-ug-net/using.html#mount-table`

7.2 Cygwin Account Names

Account names in Windows use the following format: [*authorityname*`\`]*accountname* (where *authorityname* is a computer name or domain name). The authority name isn't used in all circumstances (such as with the `BUILTIN` and `NT AUTHORITY` authorities).

Cygwin programs use the `+` rather than the `\` character to separate authority names from account names.

Cygwin's account naming rules are as follows:

- Well-known and built-in accounts will be named as in Windows, without an authority name. Examples: `Administrators` or `SYSTEM`
- If the computer **is not** a domain member, the authority name is not used. Example: `Administrator` (not `COMPUTERNAME+Administrator`)
- If the computer **is** a domain member, accounts from the computer's domain are used without an authority name. Example: `kendyer` (not `DOMAIN1+kendyer`)
- Accounts from other domains use the domain name as the authority name. Example: `DOMAIN2+lynndyer`
- Local computer accounts of a domain member computer use the computer name as the authority. Example: `COMPUTERNAME+localuser`

See the Cygwin documentation for more details about how Cygwin maps between Windows and POSIX account names.

The **Get-AccountName.ps1** script outputs one or more account names in both Windows and Cygwin formats:

- The script will output the Windows and Cygwin account names for the current account if you run it without parameters
- The script will out the Windows and Cygwin account names for one or more accounts if you provide the account name(s) as a parameter (the script also accepts pipeline input)

Run the command **Get-Help Get-AccountName** for more information about the script.

Note: OpenSSH server versions older than 8.0 required that Cygwin account names be specified using *exact case*. This requirement was removed starting in OpenSSH version 8.0.

7.3 POSIX Path Names

Cygwin programs use POSIX path names (such as `/etc/fstab`) rather than Windows path names (such as `C:\Program Files\CygSSH\etc\fstab`).

For example, the root directory (`/`) corresponds to *installpath* (e.g., `C:\Program Files\CygSSH`).

Windows drive letters (e.g., `C:`) are found in the `/cygdrive` directory (e.g., `/cygdrive/c` corresponds to `C:`).

The **cygpath** command allows you to easily translate Windows path names into a POSIX path names and vice-versa. Use the `-w` parameter to tell **cygpath** you want the Windows path.

```
Example Command               Example Output
-------------------------------------------------------------
cygpath "C:\Program Files"    /cygdrive/c/Program Files
cygpath C:\Users\KenDyer      /cygdrive/c/Users/KenDyer
cygpath -w /                  C:\Program Files\CygSSH
cygpath -w /etc/fstab         C:\Program Files\CygSSH\etc\fstab
```

In the first example, note that the quotes are required since the path name contains spaces.

If you need to specify a path name that contains spaces on a Cygwin program's command line, it may be necessary, depending on the command, to 'escape' the spaces in the path using the \ character (see section 4.1 for an example).

7.4 The Cygwin Home Directory

A user's *home directory* is the default path location for files owned by that user. By default, Cygwin maps the user's home directory to `/home/`*accountname* (see section 7.3 for more about POSIX path names).

The default can cause confusion, though, because it means that Cygwin and Windows have two separate home directory paths. For example, the Windows home directory for the `KenDyer` account might be `C:\Users\KenDyer` but the default Cygwin home directory would be something like `C:\Program Files\CygSSH\home\KenDyer`.

To avoid the confusion, the CygSSH package uses the `nsswitch.conf` file to set Cygwin's home directory to the same home directory that Windows uses. The file contains the following line:

```
db_home: /%H
```

The `%H` placeholder means 'Windows home directory in POSIX format.'

See the Cygwin documentation for more details about the `nsswitch.conf` file.

Chapter 8: Downlevel OS Version Limitations

In this documentation, operating system versions older than Windows 8.1/Server 2012 R2 are referred to as *downlevel versions*.

8.1 32-bit SSH Server on 64-bit OS Cannot Authenticate Local Account Logons

The 32-bit version of the OpenSSH server service on 64-bit downlevel OS versions cannot authenticate local account logons. This is because the **MsV1_0S4ULogon** logon type used by the OpenSSH service to impersonate local accounts is not implemented in the 32-bit emulator on 64-bit Windows OS versions until Windows 8.1/Server 2012 R2.

There are two scenarios where the above limitation does *not* apply:

- It does not apply if the computer is running a downlevel 32-bit version of Windows
- It does not apply to domain accounts if the computer is a domain member

An unfortunate implication of this limitation is that you cannot use the OpenSSH server in the following specific scenario:

- The operating system is 64-bit but not x64 (e.g., IA64), so therefore the installer installed the 32-bit version of OpenSSH
- The operating system is a downlevel OS version
- The computer is not a domain member

In this specific scenario, the OpenSSH server service will not be able to authenticate any account logons due to the limitation stated in this section.

8.2 Local Account Logons After Restart Fail Until Another Logon Occurs

The OpenSSH server service on downlevel OS versions will fail to authenticate local account logons after a restart until another logon occurs first. The installer provides the **s4ulogonfix** task (see section 1.6) that works around this problem by doing the following:

1. Creates a local user account named `MsV1_0S4ULogon` with a long, complex password
2. Grants the **Log on as a batch job** user right to the local user account
3. Creates a scheduled task named `MsV1_0S4ULogon` that causes a logon at system startup (it simply runs the Windows **hostname** command)

This workaround is only needed on downlevel OS versions if you need to authenticate using local accounts. If your computer is a domain member and you only authenticate using domain accounts, this workaround is not needed.

The workaround has two limitations:

- If the account loses the **Log on as a batch job** user right, the workaround will fail - this can occur if a Group Policy Object (GPO) in a domain configures this user right and removes the account
- If the **Network access: Do not allow storage of passwords and credentials for network authentication** security policy is set to **Enabled**, the workaround will fail - this can occur if a Group Policy Object (GPO) in a domain changes this setting to **Enabled** (it is not set to **Enabled** by default)

You can also implement the fix manually after installation if needed.

First, open an elevated Windows PowerShell console (i.e., right-click the PowerShell icon and choose `Run as administrator`). Then, run the following command:

```
Set-S4ULogonFix -Enable
```

The script will ask for confirmation before proceeding with each required configuration item.

If you specify `-Disable` rather than `-Enable` on the script's command line, the script will do the following instead:

- Revoke the **Log on as a batch job** user right from the `MsV1_0S4ULogon` local user account
- Disable the `MsV1_0S4ULogon` local user account
- Disable the `MsV1_0S4ULogon` scheduled task

Appendix A: The ChrootDirectory Setting

On a POSIX-compliant operating system (e.g., Linux), the OpenSSH `ChrootDirectory` setting uses the `chroot` system call to restrict an account's access to a specific path in the file system: The account cannot access any file or directory outside of that specific path. In POSIX terminology, this is sometimes referred to as a *chroot jail*.

The Cygwin emulation layer used by OpenSSH on Windows (see chapter 7) has a limited implementation of `chroot` because the Windows operating system does not have a direct equivalent of this system call. Consequently, Cygwin's limited implementation of `chroot` is not a true security control because it can sometimes permit the account to 'escape' from the 'jail.' (It's important to understand that even if the account 'escapes' from the 'jail,' the account is still restricted by Windows security controls; it's just that the limited Cygwin chroot jail isn't enforced as an *additional* security control.)

Even though Cygwin's implementation `chroot` is not a true security control, we can still use the `ChrootDirectory` setting in `sshd_config` to restrict the file system view for an SFTP-only account (see section 5.4).

Using the `ChrootDirectory` setting in `sshd_config` requires the following two items be in place:

1. The `passwd` file in the `etc` directory must specify that the Windows `SYSTEM` account is user number `0` (this corresponds to the POSIX `root` user account)

2. The local `sshd` account (known as the *privsep account*) must exist, and it must be **enabled** (it must *not* be disabled as stated in the OpenSSH documentation)

These two items are taken care of automatically at installation time (they don't need to be configured manually).

Appendix B: Using rsync with SSH

The **rsync** command is an efficient file and directory replication tool that supports replication over an SSH connection so that the data is encrypted in transit.

The syntax for replicating data to a remote computer using an SSH connection is as follows:

`rsync` *`options`* `-e "ssh [`*`sshoptions`*`]" "`*`source`*`" "`*`accountname@hostname`*`:`*`dest`*`"`

Where:

- *options* is one or more command line options for the **rsync** command
- *sshoptions* is one or more command line options for the **ssh** command
- *source* is the source path in POSIX format (see section 7.3); do not use \ to escape spaces in the source path
- *accountname* is the account name to connect with, in Cygwin format (see section 7.2)
- *hostname* is the remote computer name
- *dest* is the destination path in POSIX format (see section 7.3); escape spaces in the destination path using \ unless using **rsync**'s **-s** option (recommended)

Example command 1:

```
rsync -lrstvz -e "ssh" "/cygdrive/c/Users/KenDyer/Backup Files/"
  "KenDyer@computer3:/cygdrive/c/Users/KenDyer/Backup Files"
```

(All on one line) This command replicates the `C:\Users\KenDyer\Backup Files` directory to the same path on `computer3` using the `KenDyer` account.

Example command 2:

```
rsync -lrstvz -e "ssh -i /cygdrive/c/Users/filerepl/.ssh/ed25519"
  "/cygdrive/c/ProgramData/App Name/"
  "filerepl@server2:/cygdrive/c/ProgramData/App Name"
```

(All on one line) This command replicates the directory `C:\Program Data\App Name` to the computer `server2` using the `filerepl` account. Note the use of a private key file, specified as a POSIX path (see section 7.3).

Note that the source path in the example commands uses a trailing `/` to prevent creating a new directory in the destination path.

See appendix C for a link to **rsync**'s documentation.

Appendix C: Documentation Links

OpenSSH - https://www.openssh.com/manual.html

Cygwin - https://www.cygwin.com/cygwin-ug-net/cygwin-ug-net.html

rsync - https://rsync.samba.org/documentation.html

setacl - https://helgeklein/setacl/

Inno Setup - http://www.jrsoftware.org/ishelp/

Appendix D: Windows PowerShell Scripts

The following list describes the Windows PowerShell scripts provided by the CygSSH package.

Edit-SSHKey.ps1 - Updates a private key file's comment or passphrase (see section 6.2.2)

Get-AccountName.ps1 - Outputs an account's name in both Cygwin and Windows format (see section 7.2)

New-SSHKey.ps1 - Creates a key pair (see section 6.2.1)

Set-FstabConfig.ps1 - Used by the installer to create a default `fstab` file (see section 7.1)

Set-SSHGroup.ps1 - Used by the installer to create the local access group and update the `sshd_config` file (see section 5.1)*

Set-SSHHostKey.ps1 - Used by the installer to create the OpenSSH server host keys (see section 6.1)*

Set-SSHService.ps1 - Used by the installer to install, start, stop, and uninstall the OpenSSH server service*

Set-S4ULogonFix.ps1 - Used by the **s4ulogonfix** installer task (see section 1.6)**

* Only installed when setup type is **full** (see section 1.4)

** Only installed on downlevel OS versions (see section 8.2) when setup type is **full** (see section 1.4)

Note: Use the **Get-Help** cmdlet to get online help information for any of the PowerShell scripts.

Appendix E: Acknowledgments

Special thanks to the authors of the following software packages for their generous contributions:

Cygwin - https://www.cygwin.com/

OpenSSH - https://www.openssh.com/

winpty - https://github.com/rprichard/winpty/

rsync - https://rsync.samba.org/

setacl - https://helgeklein.com/setacl/

Inno Setup - http://www.jrsoftware.org/

Halibut - https://www.chiark.greenend.org.uk/~sgtatham/halibut/

PathMgr.dll - https://github.com/Bill-Stewart/PathMgr/

Appendix F: License Agreements

This package contains software from multiple sources:

- **Cygwin** - https://www.cygwin.com/licensing.html
- **OpenSSH** - https://cvsweb.openbsd.org/src/usr.bin/ssh/LICENCE?rev=HEAD
- **rsync** - https://rsync.samba.org/GPL.html
- **winpty** - https://github.com/rprichard/winpty/blob/master/LICENSE
- **setacl** - https://helgeklein.com/setacl/
- **PathMgr.dll** - https://github.com/Bill-Stewart/PathMgr/

See appendix G for source code availability for software components that require making the source code available.

Windows PowerShell Script License

In addition to the above, the package contains a set of Windows PowerShell scripts written by Bill Stewart that are used for installing and managing the software. The Windows PowerShell scripts are covered by the MIT license:

Appendix G: Open Source Source Code

Some of the software components used by the CygSSH package have a license that requires the agent (a person or organization) who distributes the software to provide the source code for the software being distributed.

To meet this requirement for the CygSSH package, the source code for the included software components covered by this kind of license is available at the following URL:

https://github.com/Bill-Stewart/CygSSH

The `/usr/src` directory contains the source code. The source code files are generally of interest only to developers. Software components not requiring source code distribution are not provided at the above URL.

Appendix H: Version History

1.6 (2021-03-05)

- Updated to OpenSSH 8.5 and mintty 3.4.6.

1.6 (2021-02-04)

- Dropped support for Windows Vista/Windows Server 2008 to increase the security of the installer.
- Updated Cygwin packages.
- Updated installer to use **PathMgr.dll** for system/user Path management.
- Updated PowerShell scripts to avoid informational messages from PowerShell parsers and to improve readibility.
- Fixed typo in **New-SSHKey.ps1** script that caused an error message "Bits has bad value 0 (too small)".
- Updated documentation to mention addition of Windows firewall rule addition and fixed some typos.

1.5

- Added missing DLLs that broke rsync.

1.4

- Updated OpenSSH to version 8.4.
- Updated rsync to version 3.2.3.
- Updated Cygwin and associated packages.
- Added default `nanorc` configuration file.

1.3

- Updated Cygwin and associated packages.
- Updated runposh tool to 1.3 (compile using FPC 3.2.0).
- Installer tweak: Clean `\usr\share` and `\usr\src` directories when installing.
- Package standardization: Installed application version matches installer version.

1.2

- Updated OpenSSH to version 8.3.
- Added **mkgroup**, **mkpasswd**, and **ssh-pageant** tools.
- Updated **runposh** tool to version 1.2.

- Added default `cygserver.conf` and `virc` configuration files.

1.1

- Updated OpenSSH service not to use the Windows ConPTY API calls in newer versions of Windows by setting the `disable_pcon` setting in the `CYGWIN` environment variable. (The CygSSH package doesn't need the ConPTY support since it uses **winpty**.) See https://cygwin.com/cygwin-ug-net/using-cygwinenv.html for details.
- Added recommendation to run **ssh** and **sftp** using **mintty**.
- Updated **runposh** tool to version 1.1.

1.0

- Initial version.

Index