NAME

ssh_config − OpenSSH SSH client configuration files

SYNOPSIS

~/.ssh/config
/etc/ssh/ssh_config

DESCRIPTION

ssh(1) obtains configuration data from the following sources in the following order:

1. command-line options
2. user's configuration file (`~/.ssh/config`)
3. system-wide configuration file (`/etc/ssh/ssh_config`)

For each parameter, the first obtained value will be used. The configuration files contain sections separated by "Host" specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

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

The configuration file has the following format:

Empty lines and lines starting with '#' are comments. Otherwise a line is of the format "keyword arguments". Configuration options may be separated by whitespace or optional whitespace and exactly one '='; the latter format is useful to avoid the need to quote whitespace when specifying configuration options using the **ssh**, **scp**, and **sftp -o** option. Arguments may optionally be enclosed in double quotes (") in order to represent arguments containing spaces.

The possible keywords and their meanings are as follows (note that keywords are case-insensitive and arguments are case-sensitive):

Host Restricts the following declarations (up to the next **Host** keyword) to be only for those hosts that match one of the patterns given after the keyword. If more than one pattern is provided, they should be separated by whitespace. A single '*' as a pattern can be used to provide global defaults for all hosts. The host is the *hostname* argument given on the command line (i.e. the name is not converted to a canonicalized host name before matching).

See **PATTERNS** for more information on patterns.

AddressFamily
Specifies which address family to use when connecting. Valid arguments are "any", "inet" (use IPv4 only), or "inet6" (use IPv6 only).

BatchMode
If set to "yes", passphrase/password querying will be disabled. This option is useful in scripts and other batch jobs where no user is present to supply the password. The argument must be "yes" or "no". The default is "no".

BindAddress
Use the specified address on the local machine as the source address of the connection. Only useful on systems with more than one address. Note that this option does not work if **UsePrivilegedPort** is set to "yes".

ChallengeResponseAuthentication
Specifies whether to use challenge-response authentication. The argument to this keyword must be "yes" or "no". The default is "yes".

CheckHostIP
If this flag is set to “yes”, ssh(1) will additionally check the host IP address in the known_hosts file. This allows ssh to detect if a host key changed due to DNS spoofing. If the option is set to “no”, the check will not be executed. The default is “yes”.

Cipher
Specifies the cipher to use for encrypting the session in protocol version 1. Currently, “blowfish”, “3des”, and “des” are supported. *des* is only supported in the ssh(1) client for interoperability with legacy protocol 1 implementations that do not support the *3des* cipher. Its use is strongly discouraged due to cryptographic weaknesses. The default is “3des”.

Ciphers
Specifies the ciphers allowed for protocol version 2 in order of preference. Multiple ciphers must be comma-separated. The supported ciphers are “3des-cbc”, “aes128-cbc”, “aes192-cbc”, “aes256-cbc”, “aes128-ctr”, “aes192-ctr”, “aes256-ctr”, “arcfour128”, “arcfour256”, “arcfour”, “blowfish-cbc”, and “cast128-cbc”. The default is:

```
aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,
aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,
aes256-cbc,arcfour
```

ClearAllForwardings
Specifies that all local, remote, and dynamic port forwardings specified in the configuration files or on the command line be cleared. This option is primarily useful when used from the ssh(1) command line to clear port forwardings set in configuration files, and is automatically set by scp(1) and sftp(1). The argument must be “yes” or “no”. The default is “no”.

Compression
Specifies whether to use compression. The argument must be “yes” or “no”. The default is “no”.

CompressionLevel
Specifies the compression level to use if compression is enabled. The argument must be an integer from 1 (fast) to 9 (slow, best). The default level is 6, which is good for most applications. The meaning of the values is the same as in gzip(1). Note that this option applies to protocol version 1 only.

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

ConnectTimeout
Specifies the timeout (in seconds) used when connecting to the SSH server, instead of using the default system TCP timeout. This value is used only when the target is down or really unreachable, not when it refuses the connection.

ControlMaster
Enables the sharing of multiple sessions over a single network connection. When set to “yes”, ssh(1) will listen for connections on a control socket specified using the **ControlPath** argument. Additional sessions can connect to this socket using the same **ControlPath** with **ControlMaster** set to “no” (the default). These sessions will try to reuse the master instance’s network connection rather than initiating new ones, but will fall back to connecting normally if the control socket does not exist, or is not listening.

Setting this to “ask” will cause ssh to listen for control connections, but require confirmation using the SSH_ASKPASS program before they are accepted (see ssh-add(1) for details). If the **ControlPath** cannot be opened, ssh will continue without connecting to a master instance.

X11 and ssh-agent(1) forwarding is supported over these multiplexed connections, however the display and agent forwarded will be the one belonging to the master connection i.e. it is not possible to forward multiple displays or agents.

Two additional options allow for opportunistic multiplexing: try to use a master connection but fall back to creating a new one if one does not already exist. These options are: “auto” and “autoask”. The latter requires confirmation like the “ask” option.

ControlPath
Specify the path to the control socket used for connection sharing as described in the **ControlMaster** section above or the string “none” to disable connection sharing. In the path, ‘%l’ will be substituted by the local host name, ‘%h’ will be substituted by the target host name, ‘%p’ the port, and ‘%r’ by the remote login username. It is recommended that any **ControlPath** used for opportunistic connection sharing include at least %h, %p, and %r. This ensures that shared connections are uniquely identified.

ControlPersist
When used in conjunction with **ControlMaster**, specifies that the master connection should remain open in the background (waiting for future client connections) after the initial client connection has been closed. If set to “no”, then the master connection will not be placed into the background, and will close as soon as the initial client connection is closed. If set to “yes”, then the master connection will remain in the background indefinitely (until killed or closed via a mechanism such as the ssh(1) “**-O** exit” option). If set to a time in seconds, or a time in any of the formats documented in sshd_config(5), then the backgrounded master connection will automatically terminate after it has remained idle (with no client connections) for the specified time.

DynamicForward
Specifies that a TCP port on the local machine be forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine.

The argument must be [*bind_address*:]*port*. IPv6 addresses can be specified by enclosing addresses in square brackets or by using an alternative syntax: [*bind_address*/]*port*. By default, the local port is bound in accordance with the **GatewayPorts** setting. However, an explicit *bind_address* may be used to bind the connection to a specific address. The *bind_address* of “localhost” indicates that the listening port be bound for local use only, while an empty address or ‘*’ indicates that the port should be available from all interfaces.

Currently the SOCKS4 and SOCKS5 protocols are supported, and ssh(1) will act as a SOCKS server. Multiple forwardings may be specified, and additional forwardings can be given on the command line. Only the superuser can forward privileged ports.

EnableSSHKeysign
Setting this option to “yes” in the global client configuration file /etc/ssh/ssh_config enables the use of the helper program ssh-keysign(8) during **HostbasedAuthentication**. The argument must be “yes” or “no”. The default is “no”. This option should be placed in the non-hostspecific section. See ssh-keysign(8) for more information.

EscapeChar
Sets the escape character (default: ‘~’). The escape character can also be set on the command line. The argument should be a single character, ‘^’ followed by a letter, or “none” to disable the escape character entirely (making the connection transparent for binary data).

ExitOnForwardFailure
Specifies whether ssh(1) should terminate the connection if it cannot set up all requested dynamic, tunnel, local, and remote port forwardings. The argument must be “yes” or “no”. The default is “no”.

ForwardAgent
Specifies whether the connection to the authentication agent (if any) will be forwarded to the remote machine. The argument must be “yes” or “no”. The default is “no”.

Agent forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the agent’s Unix-domain socket) can access the local agent through the forwarded connection. An attacker cannot obtain key material from the agent, however they can perform operations on the keys that enable them to authenticate using the identities loaded into the agent.

ForwardX11
Specifies whether X11 connections will be automatically redirected over the secure channel and `DISPLAY` set. The argument must be “yes” or “no”. The default is “no”.

X11 forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the user’s X11 authorization database) can access the local X11 display through the forwarded connection. An attacker may then be able to perform activities such as keystroke monitoring if the **ForwardX11Trusted** option is also enabled.

ForwardX11Trusted
If this option is set to “yes”, remote X11 clients will have full access to the original X11 display.

If this option is set to “no”, remote X11 clients will be considered untrusted and prevented from stealing or tampering with data belonging to trusted X11 clients. Furthermore, the `xauth`(1) token used for the session will be set to expire after 20 minutes. Remote clients will be refused access after this time.

The default is “no”.

See the X11 SECURITY extension specification for full details on the restrictions imposed on untrusted clients.

GatewayPorts
Specifies whether remote hosts are allowed to connect to local forwarded ports. By default, `ssh`(1) binds local port forwardings to the loopback address. This prevents other remote hosts from connecting to forwarded ports. **GatewayPorts** can be used to specify that ssh should bind local port forwardings to the wildcard address, thus allowing remote hosts to connect to forwarded ports. The argument must be “yes” or “no”. The default is “no”.

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

GSSAPIAuthentication
Specifies whether user authentication based on GSSAPI is allowed. The default is “no”. Note that this option applies to protocol version 2 only.

GSSAPIKeyExchange
Specifies whether key exchange based on GSSAPI may be used. When using GSSAPI key exchange the server need not have a host key. The default is “no”. Note that this option applies to protocol version 2 only.

GSSAPIClientIdentity
If set, specifies the GSSAPI client identity that ssh should use when connecting to the server. The default is unset, which means that the default identity will be used.

GSSAPIDelegateCredentials
Forward (delegate) credentials to the server. The default is “no”. Note that this option applies to protocol version 2 connections using GSSAPI.

GSSAPIRenewalForcesRekey
If set to “yes” then renewal of the client’s GSSAPI credentials will force the rekeying of the ssh connection. With a compatible server, this can delegate the renewed credentials to a session on the server. The default is “no”.

GSSAPITrustDns
Set to “yes to indicate that the DNS is trusted to securely canonicalize” the name of the host being connected to. If “no, the hostname entered on the” command line will be passed untouched to the GSSAPI library. The default is “no”. This option only applies to protocol version 2 connections using GSSAPI.

HashKnownHosts
Indicates that `ssh`(1) should hash host names and addresses when they are added to `~/.ssh/known_hosts`. These hashed names may be used normally by `ssh`(1) and `sshd`(8), but they do not reveal identifying information should the file’s contents be disclosed. The default is “no”. Note that existing names and addresses in known hosts files will not be converted automatically, but may be manually hashed using `ssh-keygen`(1).

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

HostKeyAlgorithms
Specifies the protocol version 2 host key algorithms that the client wants to use in order of preference. The default for this option is:

```
ssh-rsa-cert-v01@openssh.com,ssh-dss-cert-v01@openssh.com,
ssh-rsa-cert-v00@openssh.com,ssh-dss-cert-v00@openssh.com,
ssh-rsa,ssh-dss
```

HostKeyAlias
Specifies an alias that should be used instead of the real host name when looking up or saving the host key in the host key database files. This option is useful for tunneling SSH connections or for multiple servers running on a single host.

HostName
Specifies the real host name to log into. This can be used to specify nicknames or abbreviations for hosts. The default is the name given on the command line. Numeric IP addresses are also permitted (both on the command line and in **HostName** specifications).

IdentitiesOnly
Specifies that `ssh`(1) should only use the authentication identity files configured in the **ssh_config** files, even if `ssh-agent`(1) offers more identities. The argument to this keyword must be “yes” or “no”. This option is intended for situations where ssh-agent offers many different identities. The default is “no”.

IdentityFile
Specifies a file from which the user’s DSA, ECDSA or DSA authentication identity is read. The default is `~/.ssh/identity` for protocol version 1, and `~/.ssh/id_dsa`, `~/.ssh/id_ecdsa` and `~/.ssh/id_rsa` for protocol version 2. Additionally, any identities represented by the authentication agent will be used for authentication. `ssh`(1) will try to load cer-

tificate information from the filename obtained by appending `-cert.pub` to the path of a specified **`IdentityFile`**.

The file name may use the tilde syntax to refer to a user's home directory or one of the following escape characters: '`%d`' (local user's home directory), '`%u`' (local user name), '`%l`' (local host name), '`%h`' (remote host name) or '`%r`' (remote user name).

It is possible to have multiple identity files specified in configuration files; all these identities will be tried in sequence.

`KbdInteractiveAuthentication`
Specifies whether to use keyboard-interactive authentication. The argument to this keyword must be "yes" or "no". The default is "yes".

`KbdInteractiveDevices`
Specifies the list of methods to use in keyboard-interactive authentication. Multiple method names must be comma-separated. The default is to use the server specified list. The methods available vary depending on what the server supports. For an OpenSSH server, it may be zero or more of: "bsdauth", "pam", and "skey".

`KexAlgorithms`
Specifies the available KEX (Key Exchange) algorithms. Multiple algorithms must be comma-separated. The default is "diffie-hellman-group-exchange-sha256", "diffie-hellman-group-exchange-sha1", "diffie-hellman-group14-sha1", "diffie-hellman-group1-sha1".

`LocalCommand`
Specifies a command to execute on the local machine after successfully connecting to the server. The command string extends to the end of the line, and is executed with the user's shell. The following escape character substitutions will be performed: '`%d`' (local user's home directory), '`%h`' (remote host name), '`%l`' (local host name), '`%n`' (host name as provided on the command line), '`%p`' (remote port), '`%r`' (remote user name) or '`%u`' (local user name). This directive is ignored unless **`PermitLocalCommand`** has been enabled.

`LocalForward`
Specifies that a TCP port on the local machine be forwarded over the secure channel to the specified host and port from the remote machine. The first argument must be [*`bind_address`*:]*`port`* and the second argument must be *`host`*:*`hostport`*. IPv6 addresses can be specified by enclosing addresses in square brackets or by using an alternative syntax: [*`bind_address`*/]*`port`* and *`host`*/*`hostport`*. Multiple forwardings may be specified, and additional forwardings can be given on the command line. Only the superuser can forward privileged ports. By default, the local port is bound in accordance with the **`GatewayPorts`** setting. However, an explicit *`bind_address`* may be used to bind the connection to a specific address. The *`bind_address`* of "localhost" indicates that the listening port be bound for local use only, while an empty address or '`*`' indicates that the port should be available from all interfaces.

`LogLevel`
Gives the verbosity level that is used when logging messages from `ssh`(1). The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. The default is INFO. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of verbose output.

`MACs` Specifies the MAC (message authentication code) algorithms in order of preference. The MAC algorithm is used in protocol version 2 for data integrity protection. Multiple algorithms must be comma-separated. The default is:

```
hmac-md5,hmac-sha1,umac-64@openssh.com,
hmac-ripemd160,hmac-sha1-96,hmac-md5-96,
hmac-sha2-256,hmac-sha2-512
```

NoHostAuthenticationForLocalhost
This option can be used if the home directory is shared across machines. In this case localhost will refer to a different machine on each of the machines and the user will get many warnings about changed host keys. However, this option disables host authentication for localhost. The argument to this keyword must be “yes” or “no”. The default is to check the host key for localhost.

NumberOfPasswordPrompts
Specifies the number of password prompts before giving up. The argument to this keyword must be an integer. The default is 3.

PasswordAuthentication
Specifies whether to use password authentication. The argument to this keyword must be “yes” or “no”. The default is “yes”.

PermitLocalCommand
Allow local command execution via the **LocalCommand** option or using the **!***command* escape sequence in ssh(1). The argument must be “yes” or “no”. The default is “no”.

PKCS11Provider
Specifies which PKCS#11 provider to use. The argument to this keyword is the PKCS#11 shared libary ssh(1) should use to communicate with a PKCS#11 token used for storing the user’s private RSA key. By default, no device is specified and PKCS#11 support is not activated.

Port Specifies the port number to connect on the remote host. The default is 22.

PreferredAuthentications
Specifies the order in which the client should try protocol 2 authentication methods. This allows a client to prefer one method (e.g. **keyboard-interactive**) over another method (e.g. **password**) The default for this option is: “gssapi-with-mic, hostbased, publickey, keyboard-interactive, password”.

Protocol
Specifies the protocol versions ssh(1) should support in order of preference. The possible values are ‘1’ and ‘2’. Multiple versions must be comma-separated. The default is “2,1”. This means that ssh tries version 2 and falls back to version 1 if version 2 is not available.

ProxyCommand
Specifies the command to use to connect to the server. The command string extends to the end of the line, and is executed with the user’s shell. In the command string, ‘%h’ will be substituted by the host name to connect and ‘%p’ by the port. The command can be basically anything, and should read from its standard input and write to its standard output. It should eventually connect an sshd(8) server running on some machine, or execute **sshd -i** somewhere. Host key management will be done using the HostName of the host being connected (defaulting to the name typed by the user). Setting the command to “none” disables this option entirely. Note that **CheckHostIP** is not available for connects with a proxy command.

This directive is useful in conjunction with nc(1) and its proxy support. For example, the following directive would connect via an HTTP proxy at 192.0.2.0:

```
ProxyCommand /usr/bin/nc -X connect -x 192.0.2.0:8080 %h %p
```

PubkeyAuthentication
Specifies whether to try public key authentication. The argument to this keyword must be “yes” or “no”. The default is “yes”. This option applies to protocol version 2 only.

RekeyLimit

Specifies the maximum amount of data that may be transmitted before the session key is renegotiated. The argument is the number of bytes, with an optional suffix of 'K', 'M', or 'G' to indicate Kilobytes, Megabytes, or Gigabytes, respectively. The default is between '1G' and '4G', depending on the cipher. This option applies to protocol version 2 only.

RemoteForward

Specifies that a TCP port on the remote machine be forwarded over the secure channel to the specified host and port from the local machine. The first argument must be [*bind_address*:]*port* and the second argument must be *host*:*hostport*. IPv6 addresses can be specified by enclosing addresses in square brackets or by using an alternative syntax: [*bind_address*/]*port* and *host*/*hostport*. Multiple forwardings may be specified, and additional forwardings can be given on the command line. Privileged ports can be forwarded only when logging in as root on the remote machine.

If the *port* argument is '0', the listen port will be dynamically allocated on the server and reported to the client at run time.

If the *bind_address* is not specified, the default is to only bind to loopback addresses. If the *bind_address* is '*' or an empty string, then the forwarding is requested to listen on all interfaces. Specifying a remote *bind_address* will only succeed if the server's **GatewayPorts** option is enabled (see sshd_config(5)).

RhostsRSAAuthentication

Specifies whether to try rhosts based authentication with RSA host authentication. The argument must be "yes" or "no". The default is "no". This option applies to protocol version 1 only and requires ssh(1) to be setuid root.

RSAAuthentication

Specifies whether to try RSA authentication. The argument to this keyword must be "yes" or "no". RSA authentication will only be attempted if the identity file exists, or an authentication agent is running. The default is "yes". Note that this option applies to protocol version 1 only.

SendEnv

Specifies what variables from the local environ(7) should be sent to the server. Note that environment passing is only supported for protocol 2. The server must also support it, and the server must be configured to accept these environment variables. Refer to **AcceptEnv** in sshd_config(5) for how to configure the server. Variables are specified by name, which may contain wildcard characters. Multiple environment variables may be separated by whitespace or spread across multiple **SendEnv** directives. The default is not to send any environment variables.

See **PATTERNS** for more information on patterns.

ServerAliveCountMax

Sets the number of server alive messages (see below) which may be sent without ssh(1) receiving any messages back from the server. If this threshold is reached while server alive messages are being sent, ssh will disconnect from the server, terminating the session. It is important to note that the use of server alive messages is very different from **TCPKeepAlive** (below). The server alive messages are sent through the encrypted channel and therefore will not be spoofable. The TCP keepalive option enabled by **TCPKeepAlive** is spoofable. The server alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

The default value is 3. If, for example, **ServerAliveInterval** (see below) is set to 15 and **ServerAliveCountMax** is left at the default, if the server becomes unresponsive, ssh will disconnect after approximately 45 seconds. This option applies to protocol version 2 only.

ServerAliveInterval
Sets a timeout interval in seconds after which if no data has been received from the server, ssh(1) will send a message through the encrypted channel to request a response from the server. The default is 0, indicating that these messages will not be sent to the server. This option applies to protocol version 2 only.

StrictHostKeyChecking
If this flag is set to "yes", ssh(1) will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed. This provides maximum protection against trojan horse attacks, though it can be annoying when the /etc/ssh/ssh_known_hosts file is poorly maintained or when connections to new hosts are frequently made. This option forces the user to manually add all new hosts. If this flag is set to "no", ssh will automatically add new host keys to the user known hosts files. If this flag is set to "ask", new host keys will be added to the user known host files only after the user has confirmed that is what they really want to do, and ssh will refuse to connect to hosts whose host key has changed. The host keys of known hosts will be verified automatically in all cases. The argument must be "yes", "no", or "ask". The default is "ask".

TCPKeepAlive
Specifies whether the system should send TCP keepalive messages to the other side. If they are sent, death of the connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily, and some people find it annoying.

The default is "yes" (to send TCP keepalive messages), and the client will notice if the network goes down or the remote host dies. This is important in scripts, and many users want it too.

To disable TCP keepalive messages, the value should be set to "no".

Tunnel
Request tun(4) device forwarding between the client and the server. The argument must be "yes", "point-to-point" (layer 3), "ethernet" (layer 2), or "no". Specifying "yes" requests the default tunnel mode, which is "point-to-point". The default is "no".

TunnelDevice
Specifies the tun(4) devices to open on the client (*local_tun*) and the server (*remote_tun*).

The argument must be *local_tun*[:*remote_tun*]. The devices may be specified by numerical ID or the keyword "any", which uses the next available tunnel device. If *remote_tun* is not specified, it defaults to "any". The default is "any:any".

UsePrivilegedPort
Specifies whether to use a privileged port for outgoing connections. The argument must be "yes" or "no". The default is "no". If set to "yes", ssh(1) must be setuid root. Note that this option must be set to "yes" for **RhostsRSAAuthentication** with older servers.

User Specifies the user to log in as. This can be useful when a different user name is used on different machines. This saves the trouble of having to remember to give the user name on the command line.

UserKnownHostsFile
Specifies a file to use for the user host key database instead of ~/.ssh/known_hosts.

VerifyHostKeyDNS
Specifies whether to verify the remote key using DNS and SSHFP resource records. If this option is set to "yes", the client will implicitly trust keys that match a secure fingerprint from DNS. Insecure fingerprints will be handled as if this option was set to "ask". If this option is set to "ask", informa-

tion on fingerprint match will be displayed, but the user will still need to confirm new host keys according to the **StrictHostKeyChecking** option. The argument must be “yes”, “no”, or “ask”. The default is “no”. Note that this option applies to protocol version 2 only.

See also **VERIFYING HOST KEYS** in ssh(1).

VisualHostKey
If this flag is set to “yes”, an ASCII art representation of the remote host key fingerprint is printed in addition to the hex fingerprint string at login and for unknown host keys. If this flag is set to “no”, no fingerprint strings are printed at login and only the hex fingerprint string will be printed for unknown host keys. The default is “no”.

XAuthLocation
Specifies the full pathname of the xauth(1) program. The default is /usr/bin/xauth.

PATTERNS

A *pattern* consists of zero or more non-whitespace characters, ‘*’ (a wildcard that matches zero or more characters), or ‘?’ (a wildcard that matches exactly one character). For example, to specify a set of declarations for any host in the “.co.uk” set of domains, the following pattern could be used:

```
Host *.co.uk
```

The following pattern would match any host in the 192.168.0.[0-9] network range:

```
Host 192.168.0.?
```

A *pattern-list* is a comma-separated list of patterns. Patterns within pattern-lists may be negated by preceding them with an exclamation mark (‘!’). For example, to allow a key to be used from anywhere within an organisation except from the “dialup” pool, the following entry (in authorized_keys) could be used:

```
from="!*.dialup.example.com,*.example.com"
```

FILES

~/.ssh/config
This is the per-user configuration file. The format of this file is described above. This file is used by the SSH client. Because of the potential for abuse, this file must have strict permissions: read/write for the user, and not accessible by others.

/etc/ssh/ssh_config
Systemwide configuration file. This file provides defaults for those values that are not specified in the user’s configuration file, and for those users who do not have a configuration file. This file must be world-readable.

SEE ALSO

ssh(1)

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.