Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'ssh_config.5' command

$ man ssh_config.5

SSH_CONFIG(5) BSD File Formats Manual SSH_CONFIG(5)

NAME

ssh_config ? OpenSSH client configuration file

DESCRIPTION

ssh(1) obtains configuration data from the following sources in the fol?

lowing 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 configu?

ration 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 usually the one given on the

command line (see the CanonicalizeHostname option for exceptions).

Since the first obtained value for each parameter is used, more host-spe?

cific declarations should be given near the beginning of the file, and

general defaults at the end.

The file contains keyword-argument pairs, one per line. Lines starting

with ?#? and empty lines are interpreted as comments. Arguments may op?

tionally be enclosed in double quotes (") in order to represent arguments

containing spaces. 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 op?

tions using the ssh, scp, and sftp -o option.

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

Host Restricts the following declarations (up to the next Host or Match 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 usually the hostname argument given on the command line (see the CanonicalizeHostname keyword for excep? tions).

A pattern entry may be negated by prefixing it with an exclama? tion mark (?!?). If a negated entry is matched, then the Host entry is ignored, regardless of whether any other patterns on the line match. Negated matches are therefore useful to provide ex? ceptions for wildcard matches.

See PATTERNS for more information on patterns.

Match Restricts the following declarations (up to the next Host or Match keyword) to be used only when the conditions following the Match keyword are satisfied. Match conditions are specified us? ing one or more criteria or the single token all which always matches. The available criteria keywords are: canonical, final, exec, host, originalhost, user, and localuser. The all criteria must appear alone or immediately after canonical or final. Other criteria may be combined arbitrarily. All criteria but all, canonical, and final require an argument. Criteria may be negated by prepending an exclamation mark (?!?).

The canonical keyword matches only when the configuration file is being re-parsed after hostname canonicalization (see the CanonicalizeHostname option). This may be useful to specify con? ditions that work with canonical host names only.

The final keyword requests that the configuration be re-parsed (regardless of whether CanonicalizeHostname is enabled), and

matches only during this final pass. If CanonicalizeHostname is enabled, then canonical and final match during the same pass.

The exec keyword executes the specified command under the user's shell. If the command returns a zero exit status then the condi? tion is considered true. Commands containing whitespace charac? ters must be quoted. Arguments to exec accept the tokens de? scribed in the TOKENS section.

The other keywords' criteria must be single entries or comma-sep? arated lists and may use the wildcard and negation operators de? scribed in the PATTERNS section. The criteria for the host key? word are matched against the target hostname, after any substitu? tion by the Hostname or CanonicalizeHostname options. The originalhost keyword matches against the hostname as it was spec? ified on the command-line. The user keyword matches against the target username on the remote host. The localuser keyword matches against the name of the local user running ssh(1) (this keyword may be useful in system-wide ssh_config files).

AddKeysToAgent

Specifies whether keys should be automatically added to a running ssh-agent(1). If this option is set to yes and a key is loaded from a file, the key and its passphrase are added to the agent with the default lifetime, as if by ssh-add(1). If this option is set to ask, ssh(1) will require confirmation using the SSH_ASKPASS program before adding a key (see ssh-add(1) for de? tails). If this option is set to confirm, each use of the key must be confirmed, as if the -c option was specified to ssh-add(1). If this option is set to no, no keys are added to the agent. Alternately, this option may be specified as a time interval using the format described in the TIME FORMATS section of sshd_config(5) to specify the key's lifetime in ssh-agent(1), after which it will automatically be removed. The argument must be no (the default), yes, confirm (optionally followed by a time interval), ask or a time interval.

AddressFamily

Specifies which address family to use when connecting. Valid ar? guments are any (the default), inet (use IPv4 only), or inet6 (use IPv6 only).

BatchMode

If set to yes, user interaction such as password prompts and host key confirmation requests will be disabled. This option is use? ful in scripts and other batch jobs where no user is present to interact with ssh(1). The argument must be yes or no (the de? fault).

BindAddress

Use the specified address on the local machine as the source ad? dress of the connection. Only useful on systems with more than one address.

BindInterface

Use the address of the specified interface on the local machine as the source address of the connection.

CanonicalDomains

When CanonicalizeHostname is enabled, this option specifies the list of domain suffixes in which to search for the specified des? tination host.

CanonicalizeFallbackLocal

Specifies whether to fail with an error when hostname canonical? ization fails. The default, yes, will attempt to look up the un? qualified hostname using the system resolver's search rules. A value of no will cause ssh(1) to fail instantly if CanonicalizeHostname is enabled and the target hostname cannot be found in any of the domains specified by CanonicalDomains.

CanonicalizeHostname

Controls whether explicit hostname canonicalization is performed. The default, no, is not to perform any name rewriting and let the system resolver handle all hostname lookups. If set to yes then, for connections that do not use a ProxyCommand or ProxyJump,

ssh(1) will attempt to canonicalize the hostname specified on the command line using the CanonicalDomains suffixes and CanonicalizePermittedCNAMEs rules. If CanonicalizeHostname is set to always, then canonicalization is applied to proxied con? nections too.

If this option is enabled, then the configuration files are pro? cessed again using the new target name to pick up any new config? uration in matching Host and Match stanzas. A value of none dis? ables the use of a ProxyJump host.

CanonicalizeMaxDots

Specifies the maximum number of dot characters in a hostname be? fore canonicalization is disabled. The default, 1, allows a sin? gle dot (i.e. hostname.subdomain).

CanonicalizePermittedCNAMEs

Specifies rules to determine whether CNAMEs should be followed when canonicalizing hostnames. The rules consist of one or more arguments of source_domain_list:target_domain_list, where source_domain_list is a pattern-list of domains that may follow CNAMEs in canonicalization, and target_domain_list is a pattern-list of domains that they may resolve to.

For example, "*.a.example.com:*.b.example.com,*.c.example.com" will allow hostnames matching "*.a.example.com" to be canonical? ized to names in the "*.b.example.com" or "*.c.example.com" do? mains.

CASignatureAlgorithms

The default is handled system-wide by crypto-policies(7). Infor? mation about defaults, how to modify the defaults and how to cus? tomize existing policies with sub-policies are present in manual page update-crypto-policies(8).

Specifies which algorithms are allowed for signing of certifi? cates by certificate authorities (CAs). If the specified list begins with a ?+? character, then the specified algorithms will be appended to the default set instead of replacing them. If the

specified list begins with a ?-? character, then the specified algorithms (including wildcards) will be removed from the default set instead of replacing them.

ssh(1) will not accept host certificates signed using algorithms other than those specified.

CertificateFile

Specifies a file from which the user's certificate is read. A corresponding private key must be provided separately in order to use this certificate either from an IdentityFile directive or -i flag to ssh(1), via ssh-agent(1), or via a PKCS11Provider or SecurityKeyProvider.

Arguments to CertificateFile may use the tilde syntax to refer to a user's home directory, the tokens described in the TOKENS sec? tion and environment variables as described in the ENVIRONMENT VARIABLES section.

It is possible to have multiple certificate files specified in configuration files; these certificates will be tried in se? quence. Multiple CertificateFile directives will add to the list of certificates used for authentication.

CheckHostIP

If set to yes ssh(1) will additionally check the host IP address in the known_hosts file. This allows it to detect if a host key changed due to DNS spoofing and will add addresses of destination hosts to ~/.ssh/known_hosts in the process, regardless of the setting of StrictHostKeyChecking. If the option is set to no (the default), the check will not be executed.

Ciphers

The default is handled system-wide by crypto-policies(7). Infor? mation about defaults, how to modify the defaults and how to cus? tomize existing policies with sub-policies are present in manual page update-crypto-policies(8).

Specifies the ciphers allowed and their order of preference. Multiple ciphers must be comma-separated. If the specified list

begins with a ?+? character, then the specified ciphers will be appended to the built-in openssh default set instead of replacing them. If the specified list begins with a ?-? character, then the specified ciphers (including wildcards) will be removed from the built-in openssh default set instead of replacing them. If the specified list begins with a ?^? character, then the speci? fied ciphers will be placed at the head of the built-in openssh default set.

The supported ciphers are:

3des-cbc

aes128-cbc

aes192-cbc

aes256-cbc

aes128-ctr

aes192-ctr

aes256-ctr

aes128-gcm@openssh.com

aes256-gcm@openssh.com

chacha20-poly1305@openssh.com

The list of available ciphers may also be obtained using "ssh -Q cipher".

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 configura? tion files, and is automatically set by scp(1) and sftp(1). The argument must be yes or no (the default).

Compression

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

ConnectionAttempts

Specifies the number of tries (one per second) to make before ex?

iting. 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 timeout is applied both to establishing the connection and to performing the initial SSH protocol handshake and key ex? change.

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 connec? tion rather than initiating new ones, but will fall back to con? necting normally if the control socket does not exist, or is not listening.

Setting this to ask will cause ssh(1) to listen for control con? nections, but require confirmation using ssh-askpass(1). If the ControlPath cannot be opened, ssh(1) will continue without con? necting to a master instance.

X11 and ssh-agent(1) forwarding is supported over these multi? plexed connections, however the display and agent forwarded will be the one belonging to the master connection i.e. it is not pos? sible 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 shar? ing as described in the ControlMaster section above or the string

none to disable connection sharing. Arguments to ControlPath may use the tilde syntax to refer to a user's home directory, the to? kens described in the TOKENS section and environment variables as described in the ENVIRONMENT VARIABLES section. It is recom? mended that any ControlPath used for opportunistic connection sharing include at least %h, %p, and %r (or alternatively %C) and be placed in a directory that is not writable by other users. 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 connec? tion has been closed. If set to no (the default), then the mas? ter connection will not be placed into the background, and will close as soon as the initial client connection is closed. If set to yes or 0, then the master connection will remain in the back? ground indefinitely (until killed or closed via a mechanism such as the "ssh -O exit"). If set to a time in seconds, or a time in any of the formats documented in sshd_config(5), then the back? grounded 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. By default, the local port is bound in accordance with the GatewayPorts set? ting. 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). This option should be placed in the non-hostspecific section. See ssh-keysign(8) for more informa? tion.

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, (e.g. if either end is unable to bind and lis? ten on a specified port). Note that ExitOnForwardFailure does not apply to connections made over port forwardings and will not, for example, cause ssh(1) to exit if TCP connections to the ulti? mate forwarding destination fail. The argument must be yes or no (the default).

FingerprintHash

Specifies the hash algorithm used when displaying key finger? prints. Valid options are: md5 and sha256 (the default).

ForkAfterAuthentication

Requests ssh to go to background just before command execution. This is useful if ssh is going to ask for passwords or

passphrases, but the user wants it in the background. This im? plies the StdinNull configuration option being set to ?yes?. The recommended way to start X11 programs at a remote site is with something like ssh -f host xterm, which is the same as ssh host xterm if the ForkAfterAuthentication configuration option is set to ?yes?.

If the ExitOnForwardFailure configuration option is set to ?yes?, then a client started with the ForkAfterAuthentication configura? tion option being set to ?yes? will wait for all remote port for? wards to be successfully established before placing itself in the background. The argument to this keyword must be yes (same as the -f option) or no (the default).

ForwardAgent

Specifies whether the connection to the authentication agent (if any) will be forwarded to the remote machine. The argument may be yes, no (the default), an explicit path to an agent socket or the name of an environment variable (beginning with ?$?) in which to find the path.

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 redi? rected over the secure channel and DISPLAY set. The argument must be yes or no (the default).

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 dis? play through the forwarded connection. An attacker may then be

able to perform activities such as keystroke monitoring if the ForwardX11Trusted option is also enabled.

ForwardX11Timeout

Specify a timeout for untrusted X11 forwarding using the format described in the TIME FORMATS section of sshd_config(5). X11 connections received by ssh(1) after this time will be refused. Setting ForwardX11Timeout to zero will disable the timeout and permit X11 forwarding for the life of the connection. The de? fault is to disable untrusted X11 forwarding after twenty minutes has elapsed.

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 (the default), remote X11 clients will be considered untrusted and prevented from stealing or tam? pering with data belonging to trusted X11 clients. Furthermore, the xauth(1) token used for the session will be set to expire af? ter 20 minutes. Remote clients will be refused access after this time.

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 spec? ify 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).

GlobalKnownHostsFile

Specifies one or more files to use for the global host key data? base, separated by whitespace. The default is /etc/ssh/ssh_known_hosts, /etc/ssh/ssh_known_hosts2.

GSSAPIAuthentication

Specifies whether user authentication based on GSSAPI is allowed.

The default is no.

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.

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?.

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 will delegate the renewed credentials to a session

on the server.

Checks are made to ensure that credentials are only propagated

when the new credentials match the old ones on the originating

client and where the receiving server still has the old set in

its cache.

The default is ?no?.

For this to work GSSAPIKeyExchange needs to be enabled in the

server and also used by the client.

GSSAPIServerIdentity

If set, specifies the GSSAPI server identity that ssh should ex?

pect when connecting to the server. The default is unset, which

means that the expected GSSAPI server identity will be determined

from the target hostname.

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?.

GSSAPIKexAlgorithms

The default is handled system-wide by crypto-policies(7). Infor? mation about defaults, how to modify the defaults and how to cus? tomize existing policies with sub-policies are present in manual page update-crypto-policies(8).

The list of key exchange algorithms that are offered for GSSAPI key exchange. Possible values are

gss-gex-sha1-,

gss-group1-sha1-,

gss-group14-sha1-,

gss-group14-sha256-,

gss-group16-sha512-,

gss-nistp256-sha256-,

gss-curve25519-sha256-

This option only applies to 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 visually re? veal identifying information if the file's contents are dis? closed. The default is no. Note that existing names and ad? dresses in known hosts files will not be converted automatically, but may be manually hashed using ssh-keygen(1).

HostbasedAcceptedAlgorithms

Specifies the signature algorithms that will be used for host? based authentication as a comma-separated list of patterns. Al? ternately if the specified list begins with a ?+? character, then the specified signature algorithms will be appended to the de? fault set instead of replacing them. If the specified list be? gins with a ?-? character, then the specified signature algo? rithms (including wildcards) will be removed from the default set

instead of replacing them. If the specified list begins with a ?^? character, then the specified signature algorithms will be placed at the head of the default set. The default for this option is:

ssh-ed25519-cert-v01@openssh.com,
ecdsa-sha2-nistp256-cert-v01@openssh.com,
ecdsa-sha2-nistp384-cert-v01@openssh.com,
ecdsa-sha2-nistp521-cert-v01@openssh.com,
sk-ssh-ed25519-cert-v01@openssh.com,
sk-ecdsa-sha2-nistp256-cert-v01@openssh.com,
rsa-sha2-512-cert-v01@openssh.com,
rsa-sha2-256-cert-v01@openssh.com,
ssh-rsa-cert-v01@openssh.com,
ssh-ed25519,
ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,
sk-ssh-ed25519@openssh.com,
sk-ecdsa-sha2-nistp256@openssh.com,
rsa-sha2-512,rsa-sha2-256,ssh-rsa

The -Q option of ssh(1) may be used to list supported signature algorithms. This was formerly named HostbasedKeyTypes.

HostbasedAuthentication

Specifies whether to try rhosts based authentication with public key authentication. The argument must be yes or no (the default).

HostKeyAlgorithms

Specifies the host key signature algorithms that the client wants to use in order of preference. Alternately if the specified list begins with a ?+? character, then the specified signature algorithms will be appended to the default set instead of replacing them. If the specified list begins with a ?-? character, then the specified signature algorithms (including wildcards) will be removed from the default set instead of replacing them. If the specified list begins with a ?^? character, then the specified

signature algorithms will be placed at the head of the default set. The default for this option is:

ssh-ed25519-cert-v01@openssh.com,
ecdsa-sha2-nistp256-cert-v01@openssh.com,
ecdsa-sha2-nistp384-cert-v01@openssh.com,
ecdsa-sha2-nistp521-cert-v01@openssh.com,
sk-ssh-ed25519-cert-v01@openssh.com,
sk-ecdsa-sha2-nistp256-cert-v01@openssh.com,
rsa-sha2-512-cert-v01@openssh.com,
rsa-sha2-256-cert-v01@openssh.com,
ssh-rsa-cert-v01@openssh.com,
ssh-ed25519,
ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,
sk-ecdsa-sha2-nistp256@openssh.com,
sk-ssh-ed25519@openssh.com,
rsa-sha2-512,rsa-sha2-256,ssh-rsa

If hostkeys are known for the destination host then this default is modified to prefer their algorithms.

The list of available signature algorithms may also be obtained using "ssh -Q HostKeyAlgorithms".

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 data? base files and when validating host certificates. This option is useful for tunneling SSH connections or for multiple servers run? ning on a single host.

Hostname

Specifies the real host name to log into. This can be used to specify nicknames or abbreviations for hosts. Arguments to Hostname accept the tokens described in the TOKENS section. Nu? meric IP addresses are also permitted (both on the command line and in Hostname specifications). The default is the name given on the command line.

IdentitiesOnly

Specifies that ssh(1) should only use the configured authentica?
tion identity and certificate files (either the default files, or
those explicitly configured in the ssh_config files or passed on
the ssh(1) command-line), even if ssh-agent(1) or a
PKCS11Provider or SecurityKeyProvider offers more identities.
The argument to this keyword must be yes or no (the default).
This option is intended for situations where ssh-agent offers
many different identities.

IdentityAgent

Specifies the UNIX-domain socket used to communicate with the au?
thentication agent.
This option overrides the SSH_AUTH_SOCK environment variable and
can be used to select a specific agent. Setting the socket name
to none disables the use of an authentication agent. If the
string "SSH_AUTH_SOCK" is specified, the location of the socket
will be read from the SSH_AUTH_SOCK environment variable. Other?
wise if the specified value begins with a ?$? character, then it
will be treated as an environment variable containing the loca?
tion of the socket.
Arguments to IdentityAgent may use the tilde syntax to refer to a
user's home directory, the tokens described in the TOKENS section
and environment variables as described in the ENVIRONMENT
VARIABLES section.

IdentityFile

Specifies a file from which the user's DSA, ECDSA, authenticator-
hosted ECDSA, Ed25519, authenticator-hosted Ed25519 or RSA au?
thentication identity is read. The default is ~/.ssh/id_dsa,
~/.ssh/id_ecdsa, ~/.ssh/id_ecdsa_sk, ~/.ssh/id_ed25519,
~/.ssh/id_ed25519_sk and ~/.ssh/id_rsa. Additionally, any iden?
tities represented by the authentication agent will be used for
authentication unless IdentitiesOnly is set. If no certificates
have been explicitly specified by CertificateFile, ssh(1) will

try to load certificate information from the filename obtained by appending -cert.pub to the path of a specified IdentityFile.

Arguments to IdentityFile may use the tilde syntax to refer to a user's home directory or the tokens described in the TOKENS sec? tion.

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

Multiple IdentityFile directives will add to the list of identi? ties tried (this behaviour differs from that of other configura? tion directives).

IdentityFile may be used in conjunction with IdentitiesOnly to select which identities in an agent are offered during authenti? cation. IdentityFile may also be used in conjunction with CertificateFile in order to provide any certificate also needed for authentication with the identity.

The authentication identity can be also specified in a form of PKCS#11 URI starting with a string pkcs11:. There is supported a subset of the PKCS#11 URI as defined in RFC 7512 (implemented path arguments id, manufacturer, object, token and query argu? ments module-path and pin-value). The URI can not be in quotes.

IgnoreUnknown

Specifies a pattern-list of unknown options to be ignored if they are encountered in configuration parsing. This may be used to suppress errors if ssh_config contains options that are unrecog? nised by ssh(1). It is recommended that IgnoreUnknown be listed early in the configuration file as it will not be applied to un? known options that appear before it.

Include

Include the specified configuration file(s). Multiple pathnames may be specified and each pathname may contain glob(7) wildcards and, for user configurations, shell-like ?~? references to user home directories. Wildcards will be expanded and processed in lexical order. Files without absolute paths are assumed to be in

~/.ssh if included in a user configuration file or /etc/ssh if included from the system configuration file. Include directive may appear inside a Match or Host block to perform conditional inclusion.

IPQoS Specifies the IPv4 type-of-service or DSCP class for connections. Accepted values are af11, af12, af13, af21, af22, af23, af31, af32, af33, af41, af42, af43, cs0, cs1, cs2, cs3, cs4, cs5, cs6, cs7, ef, le, lowdelay, throughput, reliability, a numeric value, or none to use the operating system default. This option may take one or two arguments, separated by whitespace. If one argu? ment is specified, it is used as the packet class uncondition? ally. If two values are specified, the first is automatically selected for interactive sessions and the second for non-interac? tive sessions. The default is af21 (Low-Latency Data) for inter? active sessions and cs1 (Lower Effort) for non-interactive ses? sions.

KbdInteractiveAuthentication

Specifies whether to use keyboard-interactive authentication. The argument to this keyword must be yes (the default) or no. ChallengeResponseAuthentication is a deprecated alias for this.

KbdInteractiveDevices

Specifies the list of methods to use in keyboard-interactive au? thentication. 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 and pam.

KexAlgorithms

The default is handled system-wide by crypto-policies(7). Infor? mation about defaults, how to modify the defaults and how to cus? tomize existing policies with sub-policies are present in manual page update-crypto-policies(8).

Specifies the available KEX (Key Exchange) algorithms. Multiple algorithms must be comma-separated. If the specified list begins

with a ?+? character, then the specified methods will be appended to the built-in openssh default set instead of replacing them.

If the specified list begins with a ?-? character, then the specified methods (including wildcards) will be removed from the built-in openssh default set instead of replacing them. If the specified list begins with a ?^? character, then the specified methods will be placed at the head of the built-in openssh default set.

The list of available key exchange algorithms may also be obtained using "ssh -Q kex".

KnownHostsCommand

Specifies a command to use to obtain a list of host keys, in addition to those listed in UserKnownHostsFile and GlobalKnownHostsFile. This command is executed after the files have been read. It may write host key lines to standard output in identical format to the usual files (described in the VERIFYING HOST KEYS section in ssh(1)). Arguments to KnownHostsCommand accept the tokens described in the TOKENS section. The command may be invoked multiple times per connection: once when preparing the preference list of host key algorithms to use, again to obtain the host key for the requested host name and, if CheckHostIP is enabled, one more time to obtain the host key matching the server's address. If the command exits abnormally or returns a non-zero exit status then the connection is terminated.

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. Arguments to LocalCommand accept the tokens described in the TOKENS section.

The command is run synchronously and does not have access to the session of the ssh(1) that spawned it. It should not be used for

interactive commands.

This directive is ignored unless PermitLocalCommand has been en?

abled.

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 specifies the listener and may be [bind_address:]port or a Unix domain socket path. The second ar? gument is the destination and may be host:hostport or a Unix do? main socket path if the remote host supports it.

IPv6 addresses can be specified by enclosing addresses in square brackets. 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. Unix domain socket paths may use the tokens de? scribed in the TOKENS section and environment variables as de? scribed in the ENVIRONMENT VARIABLES section.

LogLevel

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

LogVerbose

Specify one or more overrides to LogLevel. An override consists of a pattern lists that matches the source file, function and line number to force detailed logging for. For example, an over? ride pattern of:

kex.c:*:1000,*:kex_exchange_identification():*,packet.c:*

would enable detailed logging for line 1000 of kex.c, everything in the kex_exchange_identification() function, and all code in the packet.c file. This option is intended for debugging and no overrides are enabled by default.

MACs The default is handled system-wide by crypto-policies(7). Infor? mation about defaults, how to modify the defaults and how to cus? tomize existing policies with sub-policies are present in manual page update-crypto-policies(8).

Specifies the MAC (message authentication code) algorithms in or? der of preference. The MAC algorithm is used for data integrity protection. Multiple algorithms must be comma-separated. If the specified list begins with a ?+? character, then the specified algorithms will be appended to the built-in openssh default set instead of replacing them. If the specified list begins with a ?-? character, then the specified algorithms (including wild? cards) will be removed from the built-in openssh default set in? stead of replacing them. If the specified list begins with a ?^? character, then the specified algorithms will be placed at the head of the built-in openssh default set.

The algorithms that contain "-etm" calculate the MAC after en? cryption (encrypt-then-mac). These are considered safer and their use recommended.

The list of available MAC algorithms may also be obtained using "ssh -Q mac".

NoHostAuthenticationForLocalhost

Disable host authentication for localhost (loopback addresses).

The argument to this keyword must be yes or no (the default).

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 (the default) or no.

PermitLocalCommand

Allow local command execution via the LocalCommand option or us? ing the !command escape sequence in ssh(1). The argument must be yes or no (the default).

PermitRemoteOpen

Specifies the destinations to which remote TCP port forwarding is permitted when RemoteForward is used as a SOCKS proxy. The for? warding specification must be one of the following forms:

PermitRemoteOpen host:port

PermitRemoteOpen IPv4_addr:port

PermitRemoteOpen [IPv6_addr]:port

Multiple forwards may be specified by separating them with white? space. An argument of any can be used to remove all restrictions and permit any forwarding requests. An argument of none can be used to prohibit all forwarding requests. The wildcard ?*? can be used for host or port to allow all hosts or ports respec? tively. Otherwise, no pattern matching or address lookups are performed on supplied names.

PKCS11Provider

Specifies which PKCS#11 provider to use or none to indicate that no provider should be used (the default). The argument to this keyword is a path to the PKCS#11 shared library ssh(1) should use to communicate with a PKCS#11 token providing keys for user au? thentication.

Port Specifies the port number to connect on the remote host. The de? fault is 22.

PreferredAuthentications

Specifies the order in which the client should try authentication methods. This allows a client to prefer one method (e.g. keyboard-interactive) over another method (e.g. password). The default is:

gssapi-with-mic,hostbased,publickey,

keyboard-interactive,password

ProxyCommand

Specifies the command to use to connect to the server. The com? mand string extends to the end of the line, and is executed using the user's shell ?exec? directive to avoid a lingering shell process.

Arguments to ProxyCommand accept the tokens described in the TOKENS section. 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 manage? ment 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

ProxyJump

Specifies one or more jump proxies as either [user@]host[:port] or an ssh URI. Multiple proxies may be separated by comma char? acters and will be visited sequentially. Setting this option will cause ssh(1) to connect to the target host by first making a ssh(1) connection to the specified ProxyJump host and then estab? lishing a TCP forwarding to the ultimate target from there. Set? ting the host to none disables this option entirely.

Note that this option will compete with the ProxyCommand option - whichever is specified first will prevent later instances of the other from taking effect.

Note also that the configuration for the destination host (either supplied via the command-line or the configuration file) is not generally applied to jump hosts. ~/.ssh/config should be used if

specific configuration is required for jump hosts.

ProxyUseFdpass

Specifies that ProxyCommand will pass a connected file descriptor back to ssh(1) instead of continuing to execute and pass data. The default is no.

PubkeyAcceptedAlgorithms

The default is handled system-wide by crypto-policies(7). Infor? mation about defaults, how to modify the defaults and how to cus? tomize existing policies with sub-policies are present in manual page update-crypto-policies(8).

Specifies the signature algorithms that will be used for public key authentication as a comma-separated list of patterns. If the specified list begins with a ?+? character, then the algorithms after it will be appended to the built-in openssh default instead of replacing it. If the specified list begins with a ?-? charac? ter, then the specified algorithms (including wildcards) will be removed from the built-in openssh default set instead of replac? ing them. If the specified list begins with a ?^? character, then the specified algorithms will be placed at the head of the built-in openssh default set.

The list of available signature algorithms may also be obtained using "ssh -Q PubkeyAcceptedAlgorithms".

PubkeyAuthentication

Specifies whether to try public key authentication. The argument to this keyword must be yes (the default) or no.

RekeyLimit

Specifies the maximum amount of data that may be transmitted be? fore the session key is renegotiated, optionally followed by a maximum amount of time that may pass before the session key is renegotiated. The first argument is specified in bytes and may have a suffix of ?K?, ?M?, or ?G? to indicate Kilobytes, Megabytes, or Gigabytes, respectively. The default is between ?1G? and ?4G?, depending on the cipher. The optional second

value is specified in seconds and may use any of the units docu? mented in the TIME FORMATS section of sshd_config(5). The de? fault value for RekeyLimit is default none, which means that rekeying is performed after the cipher's default amount of data has been sent or received and no time based rekeying is done.

RemoteCommand

Specifies a command to execute on the remote machine after suc? cessfully connecting to the server. The command string extends to the end of the line, and is executed with the user's shell. Arguments to RemoteCommand accept the tokens described in the TOKENS section.

RemoteForward

Specifies that a TCP port on the remote machine be forwarded over the secure channel. The remote port may either be forwarded to a specified host and port from the local machine, or may act as a SOCKS 4/5 proxy that allows a remote client to connect to arbi? trary destinations from the local machine. The first argument is the listening specification and may be [bind_address:]port or, if the remote host supports it, a Unix domain socket path. If for? warding to a specific destination then the second argument must be host:hostport or a Unix domain socket path, otherwise if no destination argument is specified then the remote forwarding will be established as a SOCKS proxy. When acting as a SOCKS proxy the destination of the connection can be restricted by PermitRemoteOpen.

IPv6 addresses can be specified by enclosing addresses in square brackets. 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 ma? chine. Unix domain socket paths may use the tokens described in the TOKENS section and environment variables as described in the ENVIRONMENT VARIABLES section.

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 inter? faces. Specifying a remote bind_address will only succeed if the server's GatewayPorts option is enabled (see sshd_config(5)).

RequestTTY

Specifies whether to request a pseudo-tty for the session. The argument may be one of: no (never request a TTY), yes (always re? quest a TTY when standard input is a TTY), force (always request a TTY) or auto (request a TTY when opening a login session). This option mirrors the -t and -T flags for ssh(1).

RequiredRSASize

Specifies the minimum RSA key size (in bits) that ssh(1) will ac? cept. User authentication keys smaller than this limit will be ignored. Servers that present host keys smaller than this limit will cause the connection to be terminated. The default is 1024 bits. Note that this limit may only be raised from the default.

RevokedHostKeys

Specifies revoked host public keys. Keys listed in this file will be refused for host authentication. Note that if this file does not exist or is not readable, then host authentication will be refused for all hosts. Keys may be specified as a text file, listing one public key per line, or as an OpenSSH Key Revocation List (KRL) as generated by ssh-keygen(1). For more information on KRLs, see the KEY REVOCATION LISTS section in ssh-keygen(1).

SecurityKeyProvider

Specifies a path to a library that will be used when loading any FIDO authenticator-hosted keys, overriding the default of using the built-in USB HID support.

If the specified value begins with a ?$? character, then it will be treated as an environment variable containing the path to the library.

SendEnv

Specifies what variables from the local environ(7) should be sent to the server. The server must also support it, and the server must be configured to accept these environment variables. Note that the TERM environment variable is always sent whenever a pseudo-terminal is requested as it is required by the protocol. Refer to AcceptEnv in sshd_config(5) for how to configure the server. Variables are specified by name, which may contain wild? card characters. Multiple environment variables may be separated by whitespace or spread across multiple SendEnv directives. See PATTERNS for more information on patterns.

It is possible to clear previously set SendEnv variable names by prefixing patterns with -. The default is not to send any envi? ronment variables.

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 be? ing 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 there? fore will not be spoofable. The TCP keepalive option enabled by TCPKeepAlive is spoofable. The server alive mechanism is valu? able when the client or server depend on knowing when a connec? tion has become unresponsive.

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.

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.

SessionType

May be used to either request invocation of a subsystem on the remote system, or to prevent the execution of a remote command at all. The latter is useful for just forwarding ports. The argu? ment to this keyword must be none (same as the -N option), subsystem (same as the -s option) or default (shell or command execution).

SetEnv Directly specify one or more environment variables and their con? tents to be sent to the server. Similarly to SendEnv, with the exception of the TERM variable, the server must be prepared to accept the environment variable.

StdinNull

Redirects stdin from /dev/null (actually, prevents reading from stdin). Either this or the equivalent -n option must be used when ssh is run in the background. The argument to this keyword must be yes (same as the -n option) or no (the default).

StreamLocalBindMask

Sets the octal file creation mode mask (umask) used when creating a Unix-domain socket file for local or remote port forwarding. This option is only used for port forwarding to a Unix-domain socket file.

The default value is 0177, which creates a Unix-domain socket file that is readable and writable only by the owner. Note that not all operating systems honor the file mode on Unix-domain socket files.

StreamLocalBindUnlink

Specifies whether to remove an existing Unix-domain socket file for local or remote port forwarding before creating a new one. If the socket file already exists and StreamLocalBindUnlink is not enabled, ssh will be unable to forward the port to the Unix-

domain socket file. This option is only used for port forwarding to a Unix-domain socket file.

The argument must be yes or no (the default).

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 pro? tection against man-in-the-middle (MITM) 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 ?accept-new? then ssh will automatically add new host keys to the user's known_hosts file, but will not permit connections to hosts with changed host keys. If this flag is set to ?no? or ?off?, ssh will automatically add new host keys to the user known hosts files and allow connections to hosts with changed hostkeys to proceed, subject to some restrictions. If this flag is set to ask (the default), new host keys will be added to the user known host files only after the user has con? firmed 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.

SyslogFacility

Gives the facility code that is used when logging messages from ssh(1). The possible values are: DAEMON, USER, AUTH, LOCAL0, LO? CAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The de? fault is USER.

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 tempo? rarily, 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.

See also ServerAliveInterval for protocol-level keepalives.

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 (the default). Specifying yes requests the default tunnel mode, which is point-to-point.

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 de? faults to any. The default is any:any.

UpdateHostKeys

Specifies whether ssh(1) should accept notifications of addi? tional hostkeys from the server sent after authentication has completed and add them to UserKnownHostsFile. The argument must be yes, no or ask. This option allows learning alternate hostkeys for a server and supports graceful key rotation by al? lowing a server to send replacement public keys before old ones are removed.

Additional hostkeys are only accepted if the key used to authen? ticate the host was already trusted or explicitly accepted by the user, the host was authenticated via UserKnownHostsFile (i.e. not GlobalKnownHostsFile) and the host was authenticated using a plain key and not a certificate.

UpdateHostKeys is enabled by default if the user has not overrid? den the default UserKnownHostsFile setting and has not enabled VerifyHostKeyDNS, otherwise UpdateHostKeys will be set to no.

If UpdateHostKeys is set to ask, then the user is asked to con?

firm the modifications to the known_hosts file. Confirmation is currently incompatible with ControlPersist, and will be disabled if it is enabled.

Presently, only sshd(8) from OpenSSH 6.8 and greater support the "hostkeys@openssh.com" protocol extension used to inform the client of all the server's hostkeys.

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

UserKnownHostsFile

Specifies one or more files to use for the user host key data? base, separated by whitespace. Each filename may use tilde nota? tion to refer to the user's home directory, the tokens described in the TOKENS section and environment variables as described in the ENVIRONMENT VARIABLES section. The default is ~/.ssh/known_hosts, ~/.ssh/known_hosts2.

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, information on fingerprint match will be displayed, but the user will still need to confirm new host keys according to the StrictHostKeyChecking option. The default is no.

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 finger? print string at login and for unknown host keys. If this flag is set to no (the default), no fingerprint strings are printed at login and only the fingerprint string will be printed for unknown

host keys.

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 decla? rations for any host in the ".co.uk" set of domains, the following pat? tern 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 organization except from the "dialup" pool, the following entry (in au? thorized_keys) could be used:

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

Note that a negated match will never produce a positive result by itself. For example, attempting to match "host3" against the following pattern- list will fail:

from="!host1,!host2"

The solution here is to include a term that will yield a positive match, such as a wildcard:

from="!host1,!host2,*"

TOKENS

Arguments to some keywords can make use of tokens, which are expanded at runtime:

%% A literal ?%?.

%C Hash of %l%h%p%r.

%d Local user's home directory.

%f The fingerprint of the server's host key.

%H The known_hosts hostname or address that is being searched for.

%h The remote hostname.

%I A string describing the reason for a KnownHostsCommand execution: either ADDRESS when looking up a host by address (only when CheckHostIP is enabled), HOSTNAME when searching by hostname, or ORDER when preparing the host key algorithm preference list to use for the destination host.

%i The local user ID.

%K The base64 encoded host key.

%k The host key alias if specified, otherwise the original remote hostname given on the command line.

%L The local hostname.

%l The local hostname, including the domain name.

%n The original remote hostname, as given on the command line.

%p The remote port.

%r The remote username.

%T The local tun(4) or tap(4) network interface assigned if tunnel forwarding was requested, or "NONE" otherwise.

%t The type of the server host key, e.g. ssh-ed25519.

%u The local username.

CertificateFile, ControlPath, IdentityAgent, IdentityFile, KnownHostsCommand, LocalForward, Match exec, RemoteCommand, RemoteForward, and UserKnownHostsFile accept the tokens %%, %C, %d, %h, %i, %k, %L, %l, %n, %p, %r, and %u.

KnownHostsCommand additionally accepts the tokens %f, %H, %I, %K and %t.

Hostname accepts the tokens %% and %h.

LocalCommand accepts all tokens.

ProxyCommand accepts the tokens %%, %h, %n, %p, and %r.

ENVIRONMENT VARIABLES

Arguments to some keywords can be expanded at runtime from environment variables on the client by enclosing them in ${}, for example

${HOME}/.ssh would refer to the user's .ssh directory. If a specified environment variable does not exist then an error will be returned and the setting for that keyword will be ignored.

The keywords CertificateFile, ControlPath, IdentityAgent, IdentityFile, KnownHostsCommand, and UserKnownHostsFile support environment variables.

The keywords LocalForward and RemoteForward support environment variables only for Unix domain socket paths.

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. Be? cause of the potential for abuse, this file must have strict per? missions: read/write for the user, and not writable 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 cre? ated OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

BSD August 12, 2021 BSD