HP NonStop SSH Reference Manual

HP Part Number: 544701-016

Published: February 2014
Edition: HP NonStop SSH 4.4

G06.21 and subsequent G-series RVUs
H06.07 and subsequent H-series RVUs
J06.03 and subsequent J-series RVUs

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304-1185

Contents

Preface

Who Should Read This Guide

This document is for system administrators who are responsible for installing, configuring and maintaining SSH2 components, including those delivered with the HP NonStop™ SSH product (T0801), and those that come with comForte's SecurSH or SecurFTP/SSH product.

This document also contains sections useful for users of ssh/sftp clients on NonStop systems, namely

- section “Quick Start and Guided Tour“ without sub-section “Quick-Starting the SSH2 System”
- section “SSHCOM Command Reference” (mainly regarding client mode commands)
- section “SSH and SFTP Client Reference”

Related Reading

This documentation is intended as a reference for the configuration and use of SSH components. Please also refer to additional documentation for the other products that come with the SSH2 package:

- For HP NonStop™ SSH: T0801 SOFTDOC, README or Support Notes as appropriate
- For SecurFTP: SecurFTP Quick Start Guide

The following reading is seen as prerequisite documentation for administrators installing HP NonStop™ SSH or comForte SecurSH and SecurFTP/SSH:

- HP NonStop documentation “Guardian User’s Guide”
- HP NonStop documentation “Open System Services Shell and Utilities Reference Manual”, if using OSS
- HP NonStop documentation “Guardian Procedure Errors and Messages Manual”
- HP NonStop documentation “Safeguard User’s Manual”
- HP NonStop documentation “Safeguard Administrator’s Manual”
- HP NonStop documentation “SCF Reference Manual for the Kernel Subsystem”
- HP NonStop documentation “TCP/IP Configuration and Management Manual”
- HP NonStop documentation "HP NonStop TCP/IPv6 Configuration and Management Manual"
- HP NonStop documentation “HP NonStop Cluster I/O Protocols (CIP) Configuration and Management Manual”
- HP NonStop documentation “EMS Manual”

The following reading is recommended documentation for NonStop users of SSH/SFTP clients and users connecting to NonStop using remote ssh/sftp/scp clients:

- HP NonStop documentation “Guardian User’s Guide”
- HP NonStop documentation “Open System Services Shell and Utilities Reference Manual”, if using OSS
- HP NonStop documentation “HP NonStop TACL Reference Manual”
- HP NonStop documentation “File Utility Program (FUP) Reference Manual”

Generally, users should get familiar with Guardian name space, Guardian file attributes and Guardian structured files when connecting from remote sftp/scp clients planning to transfer Guardian specific files to and from a NonStop system. This is not required if only files from and to the OSS environment will be transferred.

It is expected that administrators and users gain knowledge about the SSH standard before using SSH implementations. There are many good books about SSH. Here we only mention one:

- "SSH The Secure Shell The Definitive Guide", Daniel J. Barret et. al., O'Reilly

The following links may also serve as a starting point for SSH related information:

- http://tools.ietf.org/html/rfc4251
- http://tools.ietf.org/html/draft-ietf-secsh-filexfer-02
- http://en.wikipedia.org/wiki/Secure_Shell
- http://wiki.filezilla-project.org/SFTP_specifications
- http://www.openssh.org/

The Kerberos/GSSAPI related links shown below are of interest if Single Sign-on will be configured (see section “Single Sign-on with GSSAPI Authentication”):

- http://web.mit.edu/Kerberos/
- http://www.ietf.org/rfc/rfc4462.txt

The following reading prerequisite documentation for administrators configuring SSH2 for IPv6 support:

- HP NonStop documentation "TCP/IPv6 Migration Guide"
- HP NonStop documentation "TCP/IPv6 Configuration and Management Manual"

The following TCP/IPv6 related links may be helpful when preparing SSH2 IPv6 configuration:

- http://en.wikipedia.org/wiki/IPv6
- http://tools.ietf.org/html/rfc1639 - FTP Operation Over Big Address Records (FOOBAR)
- http://tools.ietf.org/html/rfc2428 - FTP Extensions for IPv6 and NATs
- http://tools.ietf.org/html/rfc2460 - Internet Protocol, Version 6 (IPv6) Specification
- http://tools.ietf.org/html/rfc4291 - IP Version 6 Addressing Architecture
- http://www.tcpipguide.com/free/t_IPv6Addressing.htm
- http://tools.ietf.org/html/draft-ietf-6man-text-addr-representation-04
- http://tools.ietf.org/html/rfc4038

Document History

Version 4.4

Describes changes in SSH release 97.

Documentation for the following new features has been added:

- Added STNCOM/SSHCOM OUT command and STNCOM UAIPADDR command
- Changed the range for STNCOM MAX_OPENERS, and the max continuation command length for STNCOM/SSHCOM.
- Added description for new parameter DAEMONMODEOWNERPOLICY controlling access to Daemon mode commands.
- Added description for new USER attribute OWNER allowing actions the same as defined by PARTIALSSHCOMACCESSUSER/GROUP parameters.
- Added additional information for parameter CLIENTMODEOWNERPOLICY.
- Added description for new parameters SFTPENHANCEDERRORREPORTING, PAUTHSUPPRESSIPADDRESS, HOSTKEYTYPE, HOSTKEYBITS and DNSMODE.
- Modified description for existing parameters SUBNET, INTERFACE and INTERFACEOUT.
- Added section “Multiple IP Process, Multiple IP Address Considerations” and section “TACL Subsystem and Command Interpreter Configuration”.

Changes in SSH2 release 97 that are incompatible with previous releases:

- Processing of ssh EXEC tacl requests changed in case ALLOWED-SUBSYSTEMS does not include tacl. It is now possible to execute TACL commands or macros even if tacl is not configured in ALLOWED-SUBSYSTEMS. A TACL subsystem is provided when a user gets a TACL prompt but not when just one TACL command is executed. In this way it is possible to differentiate between subsystem tacl and use of CI-PROGRAM. Previously, the execution of CI-PROGRAM via TACL command on the SSH client command line was rejected if tacl was not an allowed subsystem. The user configuration allows restricting access to TACL commands via attributes ALLOW-CI, CI-PROGRAM, CI-COMMAND and ALLOW-CI-PROGRAM-OVERRIDE to an extent that the incompatible change should not cause problems. Please see section “TACL Subsystem and Command Interpreter Configuration” and check your USER configuration accordingly for those users that do not have tacl configured in ALLOWED-SUBSYSTEMS.

Version 4.3

Describes changes in SSH2 release 96.

Documentation for the following new features has been added:

- Added additional information for parameters AUTOADDAUTHPRINCIPAL and SFTPREALPATHFILEATTRIBUTEECHOED.
- Added section "Controlling SSH and SFTP clients on NonStop via an API".
- Explained new USER attribute PTY-SERVER in section "Database for Daemon Mode".

Version 4.2

Describes changes in the SSH2 release 94.

Documentation for the following new features has been added:

- Added description for new parameters BURSTSUPPRESSION, EMSBURSTSUPPRESSION, CONSOLEBURSTSUPPRESSION, FILEBURSTSUPPRESSION, CACHEBURSTSUPPRESSION, BURSTSUPPRESSIONEXPIRATIONTIME and BURSTSUPPRESSIONMAXLOGLEVEL.

- Added additional information for parameter SHELLENVIRONMENT.
- Added additional information for authentication with password on procedure USER_AUTHENTICATE_.
- Various additions and changes in the STN Reference section.

Version 4.1

Describes changes in the SSH2 release 93.

Documentation for the following new features has been added:

- Added Migration Considerations section
- Added description of new parameter SFTPDISPLAYGUARDIAN controlling the format of filenames in SFTP informational messages.
- Added additional information displayed by the STNCOM VERSION command, and an example showing the new startup banner and version info.
- Added SSHCOM command EXPORT SSHCTL now supporting export to an OSS directory.
- Added description of additional timestamp options in utility SHOWLOG.
- Noted that macro SSH2INFO now prints warning messages if the objects SSH2, SFTPSERV and STN do not have a Safeguard DISKFILE entry with PRIV-LOGON set to ON. The warnings will also be logged at SSH2 startup.
- Added description of new STNCOM commands to provide for unique session and window name generation.
- Added description of the PROGRESS meter command option "?".
- The section "STNCOM Commands" has been updated to be in synch with STN help. New commands/parameters and EMS events for session/window naming have been added. Setmode 212 and 214 have been added in the setmode table.

Changes in SSH2 release 93 that are incompatible with previous releases:

- The STN AUTO_ADD_WIN configuration parameter is no longer supported. All openers of STN must refer to an existing window name.
- The SSHCOM STATUS SESSION brief output no longer contains the SESSION-LOG-ID field. It also now uses abbreviated column headings.

Version 4.0

Describes changes in SSH2 release 92.

Documentation for the following new features has been added:

- Added section IPv6 and description of related parameter IPMODE.
- Description for new SSH2 TCP/IP related parameters PTCPIPFILTERTCPPORTS, SOCKTCPMINRXMT, SOCKTCPMAXRXMT, SOCKTCPRXMTCNT, and SOCKTCPTOTRXMTVAL has been added.
- Added description of new SSHCOM client mode command INFO SYSTEM-USER to section "Client Mode Commands - Overview".
- Added description for new parameters LIFECYCLEPOLICYPUBLICUSERKEY, INTERVALPENDINGPUBLICUSERKEY and INTERVALLIVEPUBLICUSERKEY.
- Added description for new parameter ALLOWINFOSSH2.
- Added description for new parameters PARTIALSSHCOMACCESSGROUP<n> and PARTIALSSHCOMACCESSUSER<k>.
- Added description for new SFTP[OSS] commands append and lappend.

- Added description for new support for creation of format 2 files in an SFTP session.
- Added description for support of option -oBindAddress for SFTP[OSS] and SSH[OSS] clients.
- Added description of option LIKE for SSHCOM command ADD RESTRICTION-PROFILE.
- Updated section "Starting SSH2" with new run modes.
- Added documentation of additional commands in section "Statistics Related Commands".
- Added sections "Transfer Progress Meter" and "Controlling Transfer Summary".
- Updated section "Viewing File Contents from Guardian with SHOWLOG".
- Added description of new commands FESESSDOWN and REPLY_DELAY_MAX in section "STNCOM Commands".
- Added appendix "Event Summary".

Changes in SSH2 release 92 that are incompatible with previous releases:

- Output of SSHCOM commands that contains IP addresses in some form has been modified to allow for the greater length of IPv6 addresses

Version 3.9

Describes changes in SSH2 release 91.

Documentation for the following new features has been added:

- Added description for new parameters CPUSET and SFTPCPUSET.
- Added description for parameters AUDITEMS, AUDITFORMATCONSOLE, AUDITFORMATEMS, AUDITFORMATFILE.
- Enhanced description of SET command in section “Miscellaneous commands in SSHCOM”
- Added description for new SFTP/SFTPOSS commands FC and HISTORY.
- Added new sections “Checking SSH2 Installation”, "SSH2 License and Version Information", and "Installation of SFTPAPI".
- Added description of SSHCOM command ABORT SESSION in new section “Other Session Related Commands”.
- Added description of SSHCOM command PROMPT in section “Miscellaneous commands in SSHCOM”.

Documentation for the following already existing STN pseudo-TTY features has been added:

- Uses of STN runtime options IN/OUT.
- STNCOM: multiple line command continuation.
- Example display of INFO STN (update).
- STNCOM commands CONN_CLR_SSH, DEV_SUBTYPE, FRAGSIZE, INFO ALL, NBOT, OPENER_WAIT, PROMPT, SAVE_CFG, STNCOM_PROMPT.

Documentation for the following new STNCOM commands has been added:

- DYN_CPU (global cpu/cpu-range specification for dynamic service processes).
- NBOT_TIMEOUT

Version 3.8a

Describes changes in SSH2 release 90a.

Documentation modified for the following enhancement:

- Alphabetically sorted help items displayed within SFTP and SFTPOSS when 'help' command entered.

Version 3.8

Describes changes in SSH2 release 90.

Documentation for the following new features has been added:

- Added description for new parameters ENABLESTATISTICSATSTARTUP, INTERFACEOUT, LOGEMSKEEPCOLLECTOROPENED, LIFECYCLEPOLICYPRIVATEUSERKEY, INTERVALPENDINGPRIVATEUSERKEY and INTERVALLIVEPRIVATEUSERKEY.
- Added description for new host key related SSHCOM commands INFO HOST-KEY, EXPORT HOST-KEY
- Modified description for SSHCOM client mode commands ALTER KEY, GENERATE KEY, IMPORT KEY and INFO KEY
- Added description for new statistics related SSHCOM command STATISTICS SESSION
- Added description of new audit event SftpServerFatalErrorEvent
- Added section “FILE I/O parameters for SFTP/SFTPOSS”
- Enhanced section " Installation on the NonStop Server"
- Added an example for “Forwarding Remote Port to Local Port” in section "To Establish a Port Forwarding Tunnel with the NonStop SSH Client"

Changes in SSH2 release 90 that are incompatible with previous releases:

- In previous releases the value for INTERFACE had not been used for outgoing connections, i.e. if a TCP/IP process defined several subnets, then it was undetermined, which of the local IP addresses was used when connecting to remote systems. Now the IP address configured via INTERFACEOUT is used or, if that is not set, the value of parameter INTERFACE determines the local IP address selected for outgoing connections. The previous behavior can be activated by setting the new parameter INTERFACEOUT to value 0.0.0.0.
- The output of SSHCOM command INFO KEY has changed: The brief information contains the life-cycle state (header LIFE-CYCLE) instead of the LAST-MODIFIED field.

Version 3.7

Describes changes in SSH2 release 89.

Documentation for the following new features has been added:

- Description for SSH2 parameters ALLOWFROZENSYSTEMUSER, CLIENTMODEOWNERPOLICY and SUPPRESSCOMMENTINSSHVERSION have been added.
- Description for parameter RECORDDELIMITER now lists newly supported values CR and CRLF.
- Added description for new SSH/SFTP Client parameters SUPPRESSCLIENTBANNER, SSHERRORPREFIX, SSHINFOPREFIX and SSHQUERYPREFIX.
- Added description for new SSH/SFTP Client options -Z (corresponding to SUPPRESSCLIENTBANNER), -H (corresponding to SSHERRORPREFIX), -J (corresponding to SSHINFOPREFIX) and -K (corresponding to SSHQUERYPREFIX).
- Description of the SSH2 database was enhanced.
- Added description for new parameter SFTPEXCLUSIONMODEREAD.
- Added description of new USER attribute ALLOW-MULTIPLE-REMOTE-HOSTS
- Added section about modified behavior if an OBJECTTYPE USER record exists in Safeguard.
- Added section listing all audit messages.

- Added section for SSHCOM client mode commands RENAME KNOWNHOST and RENAME PASSWORD

Changes in SSH2 release 89 that are incompatible with previous releases:

- Previous client mode owner policy was to use the Guardian user id to store client mode records. This corresponds to value GUARDIANNAME for new parameter CLIENTMODEOWNERPOLICY. The default value for this parameter is BOTH, i.e. in order to get the previous behavior the parameter CLIENTMODEOWNERPOLICY must be explicitly set to GUARDIANNAME.
- With the introduction of parameter CLIENTMODEOWNERPOLICY it is no longer possible to execute SSHCOM GENERATE KEY for an alias if CLIENTMODEOWNERPOLICY is set to GUARDIANNAME. In previous releases this was possible although such a key had never been used (only those keys, which were stored under the Guardian id underlying an alias.
- Users that are frozen in Safeguard are no longer accepted per default (new parameter ALLOWFROZENSYSTEMUSER has default value FALSE). Previous releases allowed authentication and if that was successful (methods none, publickey and gssapi-with-mic) the user was granted access. The previous behavior can be re-established by setting parameter ALLOWFROZENSYSTEMUSER to TRUE.
- Auditing of executed SFTP commands for outgoing connections has been added. Previously there was such support for incoming connections. If an SFTP[OSS] client of release 89 or later connects via an SSH2 process of previous releases, an exception occurs (error 48) during audit initialization, i.e. an SFTP[OSS] client of release 89 or later must be used with an SSH2 process of version 89 or later.
- The AUDIT messages have been modified to include the SESSION-LOG-ID to be able to relate AUDIT messages to LOG messages and STATUS SESSION output.
- A different behavior has been implemented if an OBJECTTYPE USER record exists in Safeguard: parameter sets FULLSSHCOMACCESSGROUP<j> and FULLSSHCOMACCESSUSER<i> will be ignored.
- SUPER.SUPER no longer has full access to SSHCOM if an OBJECTTYPE USER record exists which explicitly denies SUPER.SUPER the Create authority. In previous releases SUPER.SUPER always had full access, independent of the OBJECTTYPE USER record.
- The format of audit messages has changed. Main change is the addition of the SESSION-LOG-ID at the beginning of each audit message (allowing to relate log messages and STATUS SESSION information to audit messages).
- SFTP informational messages like "Uploading ..." and "Fetching ..." now display Guardian file names in standard ssh format (Unix style with OSS prefix /G or /E) to better conform to the SFTP standard; before that, the Guardian style was the default.

Version 3.6

Describes changes in SSH2 release 88.

Documentation for the following new features has been added:

- Description for SSH2 TCP/IP related parameters SOCKETSNDBUF and SOCKETRCVBUF have been added.
- Parameter KEEPALIVE has been renamed to SOCKETKEEPALIVE.
- The "ASLINEMODE" command has been added to SFTP client commands.
- Description of newly supported SFTP transfer modes.
- Added description for new parameter SFTPEXCLUSIONMODEREAD.

Version 3.5

Describes changes in SSH2 release 87.

Documentation for the following new features has been added:

- Description for SSH2 log message memory cache related parameters LOGCACHESIZE, LOGLEVELCACHE and LOGCACHEDUMPONABORT have been added,
- Log cache related SSHCOM commands SET LOGCACHESIZE, SET LOGLEVELCACHE, SET LOGCACHEDUMPONABORT, FLUSH LOGCACHE and CLEAR LOGCACHE were described,
- Added description for SSHCOM commands STATUS SSH2, STATUS SESSION, STATUS CHANNEL and STATUS OPENER,
- The document now contains a description for file retention related SSHCOM commands ROLLOVER LOGFILE and ROLLOVER AUDITFILE.

Version 3.4

Describes changes in SSH2 release 86j.

Documentation for the following new features has been added:

- A description for SSH2 parameter ALLOWEDSUBSYSTEMS has been added,
- Parameter CLIENTALLOWEDAUTHENTICATIONS and ssh client option AllowedAuthentications has been added,
- Finer control of full SSHCOM access via SSH2 parameters FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j> are now described,
- The document now contains text about parameters SFTPEDITLINESTARTDECIMALINCR , SFTPEDITLINENUMBERDECIMALINCR and SFTPEDITLINEMODE, enhancing the control over Guardian edit lines written to NonStop (line numbers, handling of edit lines that are too long),
- Added description for parameter SFTPUPSHIFTGUARDIANFILENAMES
- SSH2 parameter STOREDPASSWORDSONLY has been described.

Version 3.3

Describes changes in SSH2 release 0086.

Documentation for the following new features has been added:

- Support of GSSAPI/Kerberos-based user authentication and key exchange in accordance with the RFC 4462 standard, including capabilities such as gssapi-with-mic, gssapi-keyex user authentication, gss-group1-sha1, and gss-gex-sha1 key exchange employing Kerberos. The new feature is addressed in new and updated documentation of the following parameters:
    - new SSH2 parameter GSSAUTH
    - new SSH2 parameter GSSKEX
    - new SSH2 parameter GSSGEXKEX
    - extended SSH2 parameter ALLOWEDAUTHENTICATIONS
    - extended USER attribute ALLOWEDAUTHENTICATIONS
    - new USER attribute PRINCIPAL
- The section "Single Sign-on with GSSAPI Authentication" has been added to the chapter "Configuring and Running SSH2"

Version 3.2

Describes changes in SSH2 release 0085.

Documentation for the following new features has been added:

- New SSH2 parameter RECORDDELIMITER

Version 3.1

Describes changes in SSH2 release 0084.

Documentation for the following new features has been added:

- New environment variable INQUIREUSERNAMEIFNOTSUPPLIED checked by ssh/sftp clients.
- New ADD USER option LIKE.
- New SSH2 parameter DISCONNECTIFUSERUNKNOWN.

Version 3.0

Describes changes in SSH2 release 0083.

Documentation for the following new features has been added:

- New database object RESTRICTION-PROFILE.
- New SSHCOM commands for manipulating of RESTRICTION-PROFILE records.
- Support for EXPORT of RESTRICTION-PROFILE records.
- New SSH2 parameter RESTRICTIONCHECKFAILEDDEFAULT.
- New USER attributes RESTRICTION-PROFILE, ALLOW-GATEWAY-PORTS, PRIORITY, COMMENT, CPU-SET and SFTP-CPU-SET.
- New attribute WIDTH for SSHCOM command EXPORT SSHCTL.
- New option FORCE for USER attributes CI-PROGRAM and SHELL-PROGRAM.
- New SSH2 parameter USETEMPLATESYSTEMUSER.

Version 2.9

Describes changes in SSH2 release 0082.

Documentation for the following new features has been added:

- Newly supported scp server functionality.
- Propagation of defines from SSH2 to shell/TACL processes started by SSH2.
- New define =SSH2^PROCESS^NAME added to shell/TACL processes started by SSH2.
- New parameter <service> after *MENU* property of USER attribute SHELL-PROGRAM.
- New USER attribute SHELL-ENVIRONMENT controlling environment for non-login shells.
- New SSH2 parameter GUARDIANATTRIBUTESEPARATOR.

A topic has been added listing the environment variables set by SSH2 when a shell is started.

Version 2.8

Describes changes in SSH2 release 0081.

Documentation for the following new features has been added:

- Documentation for new STN features: PARAM LICENSE, commands ABEND, BANNER_TIMEOUT, INPUT_TIMEOUT, IDLE_WARNING, OUTPUT_RESET, BLAST, BUFFER_SIZE, and ADD SCRIPT, and ADD SERVICE parameters RESILIENT, LIMIT, HOME, USER, LOGON, DEBUGOPT, LOGAUDIT, and SCRIPT.
- New SSHCOM commands SET AUDITFILE
- New parameter <service> after *MENU* property of USER attribute CI-PROGRAM

Version 2.7

Manual has been revised to correctly reflect the way HP NonStop SSH is delivered.

Version 2.6

Describes changes in SSH2 release 0080.

Documentation for the following new features has been added:

- Configuration of an alternate command interpreter or a service menu for USERs working with a 6530 SSH sessions
- Granting access without SSH user authentication

The chapter "STN Reference" has been added, documenting the STN pseudo TTY server.

The chapter "SFTP Client Reference" has been renamed to "SSH and SFTP Client Reference", reflecting that the chapter does now also document the SSH client program.

Version 2.5

Describes changes in SSH2 release 0074.

- Added documentation for several new SSH2 parameters: BANNER, SAFEGUARD-PASSWORD-REQUIRED, SSHAUTOKEXBYTES, SSHAUTOKEXTIME and SSHKEEPALIVETIME.
- Changes reflecting support of keyboard-interactive authentication in SSH2 DAEMON run mode.

The documentation now reflects that HP NonStop SSH is also delivered as an independent product for G-Series.

Version 2.4

The documentation now reflects that SSH2 is also delivered with the HP NonStop™ H-series release version updates (RVU) for HP Integrity NonStop™ servers (beginning with H06.11), under the product name HP NonStop SSH.

Version 2.3

Describes changes in SSH2 release 0070.

- Added section "Enabling 6530 Terminal Access" in chapter “Configuring and Running SSH2”.
- Updated Guardian SSH description in section "Secure Shell access from NonStop to Remote Systems" to reflect new capabilities.

Version 2.1

Describes changes in SSH2 releases 0062 and later.

The manual now reflects the additional functionality implemented for the SecurSH product, a complete SSH suite including shell client and server capabilities with full pseudo TTY support, as well as port forwarding. The manual contains the following major changes and additions:

- The "Installation & Quickstart" chapter has been rewritten.
- The "Configuring and Running SSH2" chapter describes additional SSH2 parameters.
- Sections for "Enabling PTY Access" and "Load Balancing" have been added.
- The "SSHCOM reference" now describes some additional USER attributes.

The following additional new features are also described:

- Running SSH2 as a NonStop process pair.
- The new mechanism for rolling over log and audit files.

Version 1.8

The new SFTP-PRIORITY attribute of user entity allows administrators to specify the priority of the SFTPSERV process started by SSH2. This feature enables SSH2 to run at a high priority, while SFTPSERV runs at a priority below other critical application or system processes. This will minimize the impact SFTP transfers have on overall system performance, while ensuring fast response times of SSH2 during SSH session establishment.

The same effect can be achieved with SFTP clients by setting the SFTP [OSS] process priority to an appropriate value.

Version 1.7

Describes changes in SSH2, releases 0044 and later:

The SFTP client now supports passwords as means as authentication. This is reflected in the following changes:

- The new entity "PASSWORD" has been added to the SSH2 user database in client mode. This is documented in the sections "SSH User Database" and "SSHCOM Command Reference".
- The Quickstart section has been updated to reflect an easier way to configure the SFTP client for a new remote host.

Version 1.6

Added description of new parameters, which allow setting of DEFINES per config file to enable configuration as a generic process:

- TCPIPHOSTFILE (sets =TCPIP^HOST^FILE)
- TCPIPNODEFILE (sets =TCPIP^NODE^FILE)
- TCPIPRESOLVERNAME (sets =TCPIP^RESOLVER^NAME)

Version 1.5

Added documentation for the PTCPIPFILTERKEY parameter.

Version 1.4

Describes changes in SSH2, release 0040. This release has the following new features:

- OSS is no longer required to run the SSH2 process.
- New SSH2 configuration parameters: SFTPPRIMARYEXTENTSIZE, SFTPSECONDARYEXTENTSIZE, SFTPMAXEXTENTS (see section "SSH2 Parameter Reference" in chapter "Configuring and Running SSH2").
- The "touch" command has been added to SFTP client commands.
- Guardian filename syntax is supported in commands working on NonStop files or subvolumes residing in the Guardian file system (see chapter "SFTP Client Reference", section "Specifying Filenames on the NonStop System").
- The attributes of files created on the NonStop system can be specified using an extended syntax in the get or put commands (see chapter "SFTP client reference", section "Extended syntax for creation of new Guardian files").

Version 1.3

Describes changes in SSH2 release 0038. This release has the following new features:

- An SFTP client to run under Guardian is supplied (see chapter "SFTP Client Reference").
- The new property SFTP-GUARDIAN-FILESET has been added to the USER property of the daemon mode database (see chapter "SSHCOM Reference").
- New commands FREEZE KEY, THAW KEY and EXPORT SSHCTL have been added to SSHCOM (see chapter "SSHCOM Reference").

Version 1.2a

- Some general improvements in layout have been implemented.
- The heading structure has been slightly revised in various places.
- Two parameters, ALLOWIP and DENYIP, have been deleted.

Version 1.2

Describes changes in SSH2 release 0036. Starting with this release, SecurFTP also supports running as an SFTP client under OSS. Documenting this new capability resulted in changes throughout the manual.

Version 1.1

Describes changes in SSH2 release 0025.

- One user now can have multiple public keys (see SSHCOM)
- New SSH2 configuration parameter: COMPRESSION
- USERBASE and USERBASEAUDIT parameters have been renamed to SSHTCL and SSHCTLAUDIT
- INFO USER command in SSHCOM now supports brief and DETAILED version of the command

Version 1.0

This is the first version of this documentation.

Introduction

The SSH2 Solution

SSH2 is a set of programs delivered when the customer purchases one of the following products:

- HP NonStop SSH. HP NonStop SSH is a comprehensive, enterprise Secure Shell solution for HP NonStop servers. In the fall of 2010, it became available from HP with the purchase of the NonStop™ Operating System Kernel for H Series and J Series NonStop platforms. For G Series releases, HP NonStop SSH continues to be available from HP as an RVU for which a license is required to obtain full functionality. For details on licensing and availability, please contact your HP Sales representative.
- comForte SecurSH. SecurSH is identical with HP NonStop SSH. It includes a remote shell and SFTP client and a shell server with full pseudo terminal support. It also offers SFTP, TCP and FTP port forwarding capabilities. The complete functionality is delivered by SSH2 programs.
- comForte SecurFTP. SecurFTP provides secure file transfer for HP NonStop systems. To protect data confidentiality across the network, it supports FTP session encryption, either via the SSL/TLS protocol (SecurFTP/SSL) or via the SSH/SFTP protocol (SecurFTP/SSH). For SecurFTP/SSH, SSH2 delivers the SFTP functionality, which is a subset of the comForte SecurSH functionality.

Fully Compliant with the SSH Protocol Specification

SSH2 is fully compliant with version 2 of the SSH (Secure Shell) protocol standard as described in various Internet draft documents (see www.ietf.org). It can be integrated with any SSH solution on UNIX, Windows or other platforms.

Strong Authentication and Multiple Cipher Suites

SSH2 supports public key authentication with key sizes of up to 2048 bits. Various ciphers, including AES and 3DES, and MACing algorithms can be selected.

Support of Full Screen Terminal Access

SSH2 supports pseudo terminals on the NonStop™ platform, allowing SSH clients to execute full screen applications, such as Emacs or vi within Secure Shell.

Built-in User Base

A built-in user base allows administrators to flexibly control who can access a system. Remote users can logon with virtual user names instead of a Guardian userid, eliminating the potential exposure of system credentials to file transfer clients. Access can be limited to a part of the file system and to a specific set of operations (e.g. only download).

Central Key Store

Instead of storing keys in the file system, SSH2 includes a key and password store with central access control, providing maximum security for user credentials. This enables the easy and secure implementation of batch processes without requiring the use of passwords in batch files.

Secure SFTP Transfer

SSH2 includes an OSS and a Guardian SFTP client, as well as an SFTP server that provides remote SFTP client access to both Guardian and OSS files. All components allow users to navigate the Guardian file system and specify files using the OSS or Guardian file name syntax, regardless of whether OSS is running. Additionally, just as with standard NonStop FTP, attributes for target files can be specified, allowing direct transfers of structured Guardian files.

TCP and FTP Port Forwarding

TCP port forwarding allows secure tunneling of Telnet sessions, as well as other connections. SSH2 also tunnels FTP sessions, securing existing FTP procedures with minimal changes. Both local and remote forwarding are supported.

Single Sign-on

SecurSH now supports user authentication and key exchange based on the GSSAPI/Kerberos 5 standards (RFC 4462). When used with a Kerberos software package on the NonStop server, this enables integration with Microsoft Active Directory and other Kerberos-based single sign-on solutions.

Note: HP does not offer a Kerberos product today, it must be purchased separately from a NonStop partner.

TCP/IPv6

Starting with version 0092 SSH2 supports IPv6 specified in RFC 2460 (Internet Protocol, Version 6). See section "TCP/IPv6 Configuration" for related configuration details and section "TCP/IPv6 Considerations" for cases specific to IPv6.

The SSH Protocol

SSH (Secure Shell), consisting of a suite of network connectivity protocols, is especially popular in UNIX environments.

SSH2 supports version 2 of the Secure Shell protocol. This version also includes specifications for a file transfer protocol. Although the name implies otherwise, this standard bears no relationship to the popular file transfer protocol known as FTP.

Components of the SSH2 Software Package

The SSH2 software package consists of the following components:

- The SSH2 component is the central component of the implementation. Depending on the mode it is started in, it can serve different purposes:
  - It implements a server process for the SSH2 protocol. It listens for incoming connections on a specific TCP/IP port (typically port 22), authenticates the user and the service and then spawns other processes it communicates with.
  - It is opened by the SSHCOM component to maintain the SSH configuration database.
  - It is opened by the SFTP or SSH client components to initiate Shell or SFTP-based file transfers to other platforms running an SSH daemon.

  The SSH2 component accesses a user database that contains the following entries for incoming SFTP connections:
  - remote user names
  - the mapping of remote user names to Guardian system users
  - user’s public keys
  - user’s credentials on the system
  - selected status information, such as the last time a user accessed the system
- The SSHOSS component implements a Secure Shell client running under OSS to connect to a remote SSH daemon. It provides Secure Shell sessions as well as TCP and FTP port forwarding capabilities.
- The SSH component implements a Secure Shell client running under Guardian to connect to a remote SSH daemon. It provides Secure Shell sessions as well as TCP and FTP port forwarding capabilities.
- The SFTPSERV component is started by SSH2 for each SFTP client that connects to SSH2 components. The SFTPSERV component then handles the file I/O associated with the file transfers initiated by the SFTP client. Because SFTPSERV is started by the SSH2 component, configuration of SFTPSERV is implicit by the configuration of the SSH2 component.
- The SFTPOSS component implements an SFTP client running under the OSS personality.
- The SFTP component implements an SFTP client running under the Guardian personality.
- The SSHCOM component allows the maintenance of the SSH user database. To do so, it communicates with the SSH2 component.
- The PAUTH component is used by SSH2 for authenticating user passwords against the system user base.
- The STN component is a pseudo TTY server providing full screen shell access to remote SSH clients.
- The SCPOSS component is the scp server implementation. It is started on request of a remote scp client via shell command. The scp client on Guardian/OSS has not been added yet.

Architecture Overview

This section shows how the various components work together in different usage scenarios.

SSH2 Running as SSH Daemon (Server)

The following figure shows how the components of SSH2 work together to implement SSH server processes (often referred to as a “daemon” in UNIX environments) on the NonStop system. These SSH processes provide shell, file transfer and port forwarding access to remote SSH clients, such as OpenSSH on UNIX:

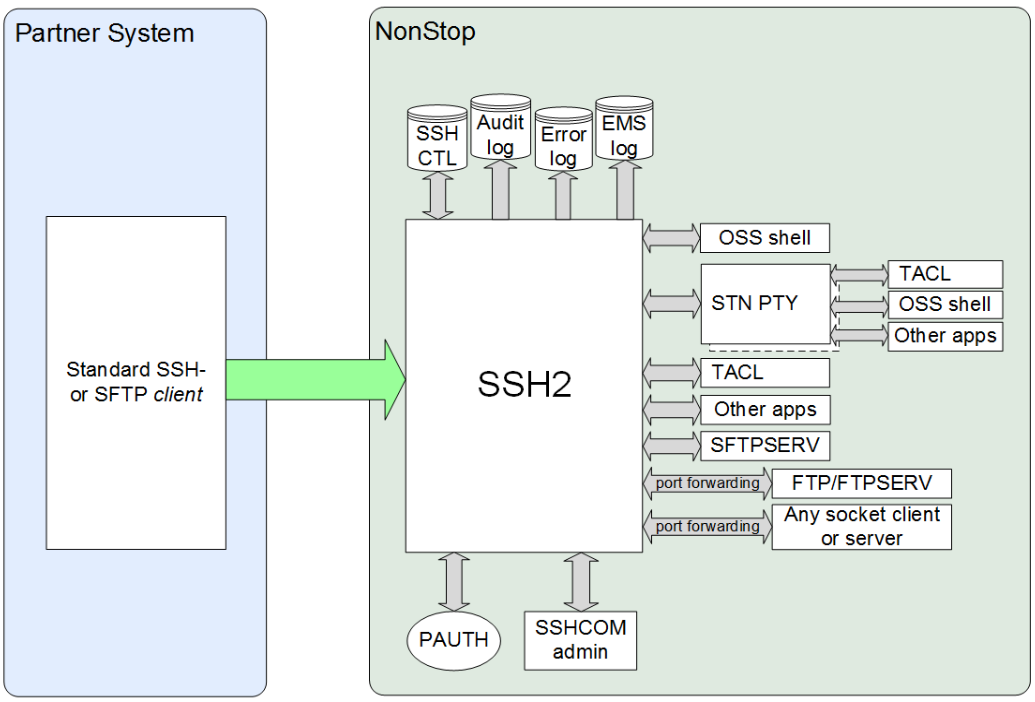


Figure 1: SSH2 running as SSH daemon

The SSH2 component accepts the incoming TCP/IP session and authenticates the remote user against the user database, optionally verifying user passwords with the PAUTH process. Upon request it …

- spawns an OSS shell, TACL or SFTPSERV process.
- allocates a PTY (a pseudo terminal) by communicating to an STN process acting as a PTY server.
- forwards TCP/IP or FTP connections from the remote SSH client to a local server process or vice versa.

The SSHCOM component is used to maintain the user database, allowing administrators to configure remote user's public keys and control access rights to server functionality and the file system for file transfer.

SSH2 Running as SSH Client

The following figure shows how the components of SSH2 work together to implement an SSH client running on the NonStop platform:

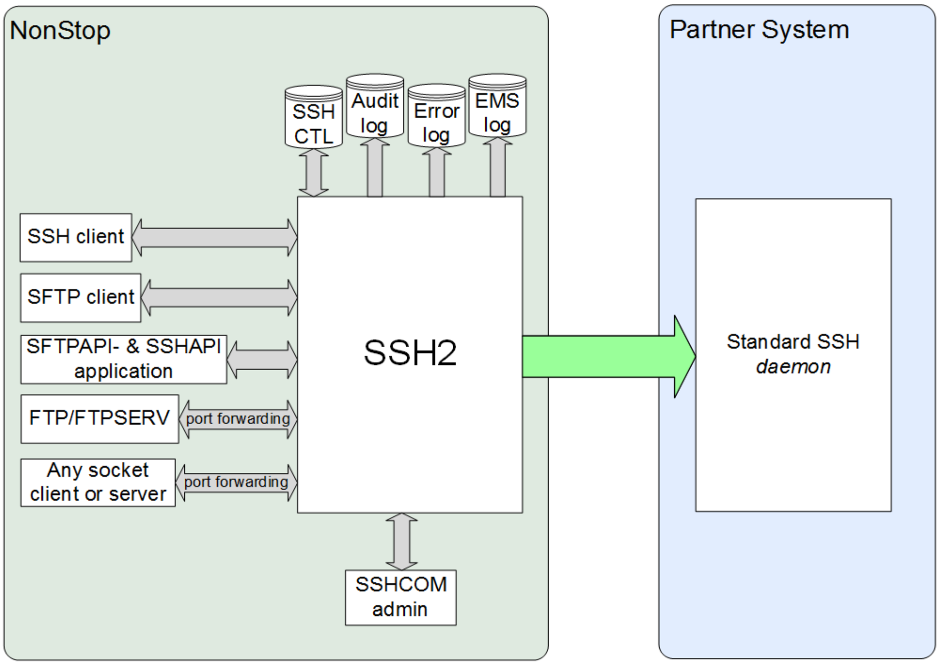


Figure 2: SSH2 running as SSH client

SSH2 can interface with a range of client components, including SSH, SFTP or the equivalent OSS programs, such as SSHOSS or SFTPOSS. With SSH2, a client component opens the SSH2 component and forwards the user commands and the startup configuration.

Applications can establish outgoing SSH or SFTP sessions using SFTPAPI or SSHAPI, see section "Controlling SSH and SFTP Clients on NonStop™ via an API".

The SSH2 component connects to the remote system via TCP/IP and does the setup of the SSH session. The client component and the SSH2 component keep exchanging messages via $RECEIVE until the client is terminated by the user.

Additionally, a client can establish port forwarding to forward TCP/IP or FTP connections from local socket programs to the remote SSH server or vice versa.

The SSHCOM component is used to maintain the key store containing the local system user's key pairs, remote passwords and remote SSH host's public keys.

Installation & Quick Start

System Requirements

To run SSH2 components, associated systems must meet the following requirements:

HP NonStop™ host:

- G-Series: G06.21 or later.
- H-Series: H06.07 or later.
- J-Series: J06.03 or later
- OSS is **not** required. If present, OSS is fully supported.

Partner systems:

- An SSH client and/or daemon supporting version 2 of the SSH protocol.

Acquiring the Product Archives

The HP NonStop SSH product is delivered with the H-series Release Version Update (RVU) H06.11 and later, or the J-series RVU J06.03 and later. A license file is no longer required for H06.21 and later, or J06.10 and later. These releases correspond to SPR T0801AAQ and later. For G06.32 and G06.32 based Time Critical Fix releases (TCFs), NonStop SSH is only licensed for use with MR-Win6530 on the NonStop System Console (NSC) for secure communications with the default IP maintenance stacks. To enable full product use you must contact your HP Sales representative for details on licensing.

SSH2 also comes with the comForte SecurSH or SecurFTP/SSH product packages. These products require the SSH2 installation archive (SSHINST.100 or SSHINST.800, depending on the NonStop Server type) to be unpacked on the NonStop server.

Installation on the NonStop™ Server

Note(s):

- For SSH2 as part of HP NonStop SSH, the installation procedures are different and the steps outlined in sections "Installing the SSH Components on the NonStop System" and "Quick-starting the SSH2 System" should be skipped.
- HP NonStop SSH will be pre-installed with your H-series RVU, J-series RVU, or G-series RVU (G06.32 or later). This enables SSH connectivity on the default TCP/IP stacks. Please refer to the SOFTDOC and support notes details for information on enabling SSH on additional TCP/IP stacks.
- For G-Series prior to G06.32, perform the standard independent product installation procedure and refer to the README file for post-installation instructions.
  Both for H-Series and G-Series, the installation subvolume of HP NonStop SSH is $SYSTEM.ZSSH and the processes are managed through the SCF Kernel manager $ZZKRN
- As of H06.22/J06.11 (SPR T0801^AAS), a configuration file named SSHMCFG has been added for exclusive use by SSH2 processes $ZSSP0 and $ZSSP1 configured for the maintenance LANs. SSHMCFG has entries specifying a dedicated data base SSHMDB, a dedicated host key file HOSTKEYM and log file SSHMLOG. The original SSHCFG file can now be used for SSH2 processes configured for non-maintenance LANs, but keep in mind that this file will be overwritten with the installation of a new RVU. A backup should be kept in case changes have been made.
- Note that if for some reason the installation subvolume is chosen to be other than $SYSTEM.ZSSH, the startup files (ZSSHGP, SSHCFG, SSHMCFG) must be changed to point to the correct locations. Therefore, it is recommended to keep the production installation always in $SYSTEM.ZSSH. The executables SSH2 (SSH server) and STN (pseudo-TTY) reside in this subvolume as well, they are not placed in $SYSTEM.SYSnn; however, the executables SSHCOM, SSH, and SFTP are installed in $SYSTEM.SYSnn.
- The startup parameter for processes $ZSSP0 and $ZSSP1 has been modified in the ZSSHGP file for SPR T0801^AAS, and now points to configuration file SSHMCFG instead of SSHCFG in the ADD process section and a new ALTER process section. After a fallback to a pre-J06.11/H06.22 RVU or to an SPR prior to T0801^AAS, the $ZSSP0 and $ZSSP1 processes will not start because their startup parameter definition points to configuration file SSHMCFG which does not exist in pre-AAS NonStop SSH releases. The ZSSHGP file in earlier NonStop SSH releases does not contain an ALTER section, and the "process add" commands in the ADD section fail because the process definitions already exist. To resolve this problem, issue these commands at a TACL prompt:
  SCF DELETE PROCESS $ZZKRN.#SSH-ZTCP*
  RUN ZMODGP $SYSTEM.ZSSH.ZSSHGP
  SCF START PROCESS $ZZKRN.#SSH-ZTCP*

Installing the SSH Components on the NonStop™ System

After you have downloaded the files to your workstation, transfer the SSH2 installation archive (SSH2INS.100 or SSHINSTI.800, depending on the NonStop Server type) to your NonStop system, alter the file code and run the installation program.

1. Using your favorite file transfer program, transfer the SSH installation archive (SSHINST.100 or SSHINST.800) in binary mode to your NonStop system. Copy the file to the subvolume on which you want to install the components.
2. Alter the installation archive file code. On G-series:

```
FUP ALTER SSHINST, CODE 100
```

On H- and J-Series

```
FUP ALTER SSHINST, CODE 800
```

3. Extract the archive by issuing the following command:

```
RUN SSHINST
```

The SSH program files will now be copied to the assigned subvolume.

4. For the Safeguard versions T9750G07^AFO/T9750H04^AFJ and later set the PRIV-LOGON bit for objects SSH2, SFTPSERV and STN (if not already executed by DSM/SCM), e.g.:

```
SAFECOM ADD DISKFILE $SYSTEM.ZSSH.SSH2, PRIV-LOGON ON
SAFECOM ADD DISKFILE $SYSTEM.ZSSH.SFTPSERV, PRIV-LOGON ON
SAFECOM ADD DISKFILE $SYSTEM.ZSSH.STN, PRIV-LOGON ON
```

Note: Note: Macro SSH2INFO prints warning messages if the objects SSH2, SFTPSERV and STN do not have a Safeguard DISKFILE entry with PRIV-LOGON set to ON.
The SSH2 process now also checks at startup if those objects have a Safeguard DISKFILE entry with PRIV-LOGON set to ON. If this is not the case, then a warning will be logged. Without PRIV-LOGON ON, the mentioned processes may not be able to impersonate other users correctly (needed after authentication). Not setting PRIV-LOGON may also cause delays leading to interruption of service.

Unlocking the Product with a License File

If you did not purchase NonStop SSH with the NonStop™ Operating System Kernel for H Series and J Series NonStop platforms, you will need a license file to use SSH components. The license file is tied to your system number.

The license file should be called LICENSE (which is the default name if not otherwise specified using the license parameter) and should reside on the same subvolume as the SSH2 component. If you need to put the license file in a different location you must use the PARAMETER LICENSE to specify the location. If there is a problem with the license file, the SSH2 component will issue a message on startup and terminate.

If the license file is valid you will see the expiration date in a log message during startup.

Note: For HP NonStop SSH on S-Series or if you did not purchase NonStop SSH with the NonStop Operating System Kernel for H Series and J Series, the default SSH installation restricts the use of the product to the MR-Win6530 terminal emulator client running on a NonStop System Console, and also restricts the use of the product to certain HP tools, such as HP Systems Insight Manager. These tools use a special key to invoke the SSH client. To unlock functionality for general use, you will need to request a license file from HP. Send an email to license.manager@hp.com and include customer name, system id, system type, and the date when the order for the software was placed.

SSH2 License and Version Information

The SSH2 release provides a TACL macro that retrieves license and version information. After changing the current subvolume to a subvolume containing an SSH2 installation the macro is started using the RUN command, e.g.:

```
VOLUME $SYSTEM.ZSSH
RUN SSH2INFO
```

The SSH2INFO macro will display the content of the license file (if found). First the default subvolume will be checked when looking for the license file, then the standard installation subvolume $SYSTEM.ZSSH.

Then the macro lists the vproc information of the files SSH2, SFTPSERV, SFTP, SFTPOSS, SSH, SSHOSS, SSHCOM, SCPOSS, STN and SHOWLOG.

For objects SFTP, SSH, SSHCOM, SHOWLOG the macro checks the default subvolume first, then subvolume $SYSTEM.SYSnn and finally $SYSTEM.ZSSH. The vproc information of all objects found is retrieved but only the vproc of the first object found is displayed. These objects are expected to reside in subvolume $SYSTEM.SYSnn after the standard HP installation process.

For the other objects, namely SSH2, SFTPSERV, SFTPOSS, SSHOSS, SCPOSS, STN, the SSH2INFO macro checks the default subvolume first, then subvolume $SYSTEM.ZSSH and finally $SYSTEM.SYSnn. The vproc information of all objects found is retrieved but only the vproc of the first object found is displayed. These objects are expected to reside in subvolume $SYSTEM.ZSSH after the standard HP installation process.

The retrieved vprocs are then used to execute a consistency check: A warning will be issued if an object exists in both locations $SYSTEM.ZSSH and $SYSTEM.SYSnn and the vproc information differs.

Updating to a new version of the SSH2 file set

The following describes how to upgrade to a new version of SSH2 and its related object files. It assumes that an older version of the product is already running successfully and configured correctly.

Download of the object file set

1. Download from the comForte web site:

   As first step please download the PAK archive containing the new files from the comForte web site. This will be a single file with an extension ".100" for S-Series and extension ".800" for H-Series.

2. Transfer file to NonStop system and unPAK in scratch subvolume:

   Transfer the file to the NonStop system in binary and FUP ALTER it to the file code 100/800 as indicated by the extension. RUN the file and the new object files will be placed on the scratch subvolume.

Installation of the new version

1. Backup your existing object files.
2. Stop all SSH2 instances.

   It is assumed that you have a standard way to STOP all running SSH2 instances.

3. FUP DUP the new object files from the scratch subvolume to your production subvolume.
4. Restart the SSH2 instances with the new version.

   It is assumed that you have a standard way to restart the SSH2 processes.

This is the time to check that the new version of SSH2 is running properly in your environment.

5. Backing out the new version.

In case the new version of SSH2 creates unexpected problems, revert to the old object files.

Where configuration data is stored

Other than any macros you have created, there are two data files which you want to keep in order to keep your existing database/configuration entries: HOSTKEY stores the host key SSHCTL stores all users and configuration done through SSHCOM.

Migration Considerations

When migrating from one NSK system to another, the original configuration can be preserved by porting the SHCTL database, the HOSTKEY file, and the SSH configuration file to the SSH subvolume $system.zssh. The migration should only be done for SSH2 processes associated with non-maintenance LANs. Note that the configuration file SSHCFG is a template and will be overwritten by DSM/SCM when a new SPR is installed. Therefore, the ported configuration file should be named differently, and the startup message in the SCF input file for persistent processes (or the startup obey file) changed to point to the correct configuration file. Also take note that if a license file existed in the original configuration, but not required any longer in the target system (SPRs >= T0801^AAQ), the customer name from the license file must be placed as a value for parameter CUSTOMER in the configuration file.

Installation of SFTPAPI

SFTPAPI is a separately licensed module offering a programmatic interface to SFTP similar to FTPAPI for FTP.

In June, 2011, HP started to offer the SFTPAPI product which requires a special license. It enables users to easily convert existing FTP scripts/programs to switch over to SFTP. The minimum SPR supporting this feature is T0801^AAQ for H/J series, and T0801^AAT for G-series. The HP NonStop SFTP API Reference Manual, part number 659755-nnn, describes the API in detail. Support for it is built into the SFTP client, which must be placed together with the license into a dedicated subvolume.

Currently it is not possible to use the SSH home subvolume $SYSTEM.ZSSH because of conflicts in the license naming and license checking. To simplify the installation process, starting with TCF T0801^AAY (H/J series) and T0801^AAZ (G-series), the SFTP client will be distributed in $SYSTEM.SYSnn (as before) and in $SYSTEM.ZSFTPAPI. The user needs to place the SFTPAPI license (named "LICENSE") into the $SYSTEM.ZSFTPAPI subvolume where the additional copy of the SFTP object is located. In the program that makes the FTP API calls, the variable FTPPGM pointing to the FTP client must be modified to point to the SFTP client $SYSTEM.ZSFTPAPI.SFTP.

Quick Start and Guided Tour

This section offers a brief example illustrating how to start SSH2. In addition, we will provide a guided tour that illustrates how to perform various SSH related tasks with a remote SSH system.

We will base this section on some assumptions:

- OpenSSH is installed on the remote system, with sshd listening on port 22.
- The IP address of the NonStop system is 10.0.0.199.
- The IP address of the remote system is 10.0.0.201.
- The SSH2 server will listen on port 22

Some of the steps illustrated here are only covered briefly; however these steps are covered in detail in subsequent sections of this documentation.

Quick-Starting the SSH2 System

This section illustrates how to quickly start the SSH2 system and provides an overview of the functionality available. For production installation, you will need to consider availability, load balancing and security related issues. Please refer to the "Configuring and Running SSH2" chapter for details.

To start the STN Pseudo Terminal Server

To enable remote SSH clients to allocate a pseudo terminal for full screen access, you will need to start an STN process to act as a PTY server for SSH2. You may omit this step if full screen access is not required.

1. At the TACL prompt, issue the following commands:

```
CLEAR ALL PARAM
PARAM BACKUPCPU ANY
RUN STN/NAME $PTY, NOWAIT/
```

2. Verify if the process started successfully by checking its status and EMS for any error messages.

Note: For productive use of the STN component, it is recommended that you install the EMS template file ZSTNTMPL using standard installation procedures. This will ensure that STN EMS messages will be displayed correctly.

To Start the SSH2 Component

Note: The SSH2 process must be started and run under the SUPER.SUPER logon. When started using a different user ID, the process will issue a warning message and terminate.

1. SSH2 can be started easily. At the TACL prompt, issue the following commands:

```
CLEAR ALL PARAM

RUN SSH2/NAME $SSH01, CPU 1/ ALL; &

PORT 22; &

AUTOADDSYSTEMUSERS true; &

ALLOWTCPFORWARDING true; &

STRICTHOSTKEYCHECKING false
```

Following are details on these instructions:

- "$SSH01" is the process name of the SSH2 process. Setting the process name to "$SSHnn"—with nn being the number of the CPU in which SSH2 is started—will allow the NonStop SSH and SFTP clients to automatically find the SSH2 process handling the SSH protocol layer for them.
- In a production environment it is recommended to specify run option NOWAIT as well as run options TERM and OUT with a virtual home terminal as value, e.g. TERM $ZHOME, OUT $ZHOME (Please replace $ZHOME with $VHS or other process name as needed.) When you start SSH2 in NOWAIT mode, make sure you have disabled logging to the home terminal. To do so, set PARAM LOGCONSOLE *.
- The keyword "ALL" designates that the SSH2 component will be allowing all supported functionality. (For more information, see chapter "Configuring and Running SSH2" for details on the run modes of the SSH2 process.)
- The parameter "PORT" reflects the port number SSH2 will listen on for incoming SSH connections.
- The parameter "AUTOADDSYSTEMUSERS" controls whether remote users can log on via SSH using a Guardian user ID or alias, without configuring them explicitly via SSHCOM in the SSHCTL.
- The parameter "ALLOWTCPFORWARDING" controls whether port forwarding is generally allowed.
- The parameter "STRICTHOSTKEYCHECKING" controls whether client access to remote systems is limited to hosts with their public key explicitly configured as a KNOWNHOST entity in the SSHCTL.

With this parameter set to false, users will be prompted if they want to continue a connection to an unknown host.

Note: When you start SSH2 in NOWAIT mode, make sure you have disabled logging to the home terminal. To do so, set the following PARAM:
PARAM LOGCONSOLE *

2. SSH2 will now start with the parameters specified in the command line. It will output initialization messages to your terminal. Please check these messages for any errors.

Note: Set the DEFINE =TCPIP^PROCESS^NAME or the parameter SUBNET accordingly if you want to run SSH2 over a TCP/IP process other than $ZTC0.

Upon first startup, SSH2 will create a HOSTKEY for the DAEMON mode, which may take a few seconds, depending on the speed of your system. SSH2 will also create the SSHCTL configuration data base.

Note: If you have installed SSH2 on a non-audited disk volume, SSH2 will fail to open the SSHCTL with error 80 (Invalid operation on audited file or non-audited disk volume). For testing, you may add SSHCTLAUDIT FALSE to the startup parameters to work around this problem. For a production installation, however, it is strongly recommended that you have SSHCTL audited. Use the SSHCTL parameter to specify a filename on an audited disk volume, if required.

A normal startup output looks similar to the following screen shot:

```
$SSH01|20Jan14 15:34:01.52|20|---------------------------------------------------------
$SSH01|20Jan14 15:34:01.52|10|SSH2 version T9999H06_22Jan2014_comForte_SSH2_0097
$SSH01|20Jan14 15:34:01.53|10|config file:   '$QAHPSSH.T0801ABK.ztc1cfg'
$SSH01|20Jan14 15:34:01.53|10|config2 file:  '*'
$SSH01|20Jan14 15:34:01.54|20|object filename is  '\BWNS02.$QAHPSSH.T0801ABK.SSH2'
$SSH01|20Jan14 15:34:01.54|20|object subvolume is '\BWNS02.$QAHPSSH.T0801ABK',
priority is 150
$SSH01|20Jan14 15:34:01.54|20|dumping configuration:
[file ] *                     <log configuration>
[def  ] ALLOWEDAUTHENTICATIONS <keyboard-interactive,password,publickey>
[file ] ALLOWEDSUBSYSTEMS     <sftp,tacl>
[def  ] ALLOWFROZENSYSTEMUSER <FALSE>
[def  ] ALLOWINFOSSH2         <ALL>
[def  ] ALLOWPASSWORDSTORE    <TRUE>
[file ] ALLOWTCPFORWARDING    <TRUE>
[def  ] AUDITCONSOLE          <*>
[file ] AUDITFILE             <$QAHPSSH.T0801ABK.ZTC1AUD>
[file ] AUDITFILERETENTION    <10>
[def  ] AUDITFORMAT           <21>
[file ] AUDITMAXFILELENGTH    <1000>
[def  ] AUTOADDAUTHPRINCIPAL <FALSE>
[file ] AUTOADDSYSTEMUSERS    <TRUE>
[def  ] BACKUPCPU             <NONE>
[def  ] BANNER                <*>
[def  ] BURSTSUPPRESSION      <FALSE>
[def  ] BURSTSUPPRESSIONEXPIRATIONTIME <300>
[def  ] BURSTSUPPRESSIONMAXLOGLEVEL <40>
[def  ] CACHEBURSTSUPPRESSION <FALSE>
[def  ] CIPCOMPATERROR        <*>
[def  ] CIPHERS               <aes256-cbc,twofish256-cbc,twofish-cbc,aes128-
cbc,twofish128-cbc,blowfish-cbc,3des-cbc,arcfour,cast128-cbc>
[def  ] CLIENTALLOWEDAUTHENTICATIONS <none,gssapi-with-
mic,publickey,password,keyboard-interactive>
[file ] CLIENTMODEOWNERPOLICY <GUARDIANNAME>
[def  ] COMPRESSION           <TRUE>
[run  ] CONFIG                <$QAHPSSH.T0801ABK.ztc1cfg>
[def  ] CONFIG2               <*>
[def  ] CONSOLEBURSTSUPPRESSION <FALSE>
[def  ] CPUSET                <>
[def  ] CUSTOMER              <>
[file ] DAEMONMODEOWNERPOLICY <LOGINNAME>
[def  ] DNSMODE               <FIRST>
[def  ] EMSBURSTSUPPRESSION   <FALSE>
```

```
[def  ] ENABLESTATISTICSATSTARTUP <FALSE>
[def  ] FILEBURSTSUPPRESSION <FALSE>
[def  ] FULLSSHCOMACCESSGROUP1 <>
[def  ] FULLSSHCOMACCESSUSER1 <>
[def  ] GSSAUTH              <*>
[def  ] GSSGEXKEX            <FALSE>
[def  ] GSSKEX               <TRUE>
[def  ] GUARDIANATTRIBUTESEPARATOR <,>
[def  ] HOSTKEY              <HOSTKEY>
[def  ] HOSTKEYBITS          <1024>
[def  ] HOSTKEYTYPE          <DSA>
[def  ] INTERFACE            <0.0.0.0>
[expl ] INTERFACEOUT         <0.0.0.0>
[def  ] INTERVALLIVEPRIVATEUSERKEY <730>
[def  ] INTERVALLIVEPUBLICUSERKEY <730>
[def  ] INTERVALPENDINGPRIVATEUSERKEY <0>
[def  ] INTERVALPENDINGPUBLICUSERKEY <0>
[def  ] IPMODE               <IPV4>
[def  ] LICENSE              <\BWNS02.$QAHPSSH.T0801ABK.LICENSE>
[def  ] LIFECYCLEPOLICYPRIVATEUSERKEY <DISABLED>
[def  ] LIFECYCLEPOLICYPUBLICUSERKEY <DISABLED>
[def  ] LOGCACHEDUMPONABORT  <TRUE>
[def  ] LOGCACHESIZE         <1024>
[file ] LOGCONSOLE           <*>
[file ] LOGEMS               <*>
[def  ] LOGEMSKEEPCOLLECTOROPENED <TRUE>
[file ] LOGFILE              <$QAHPSSH.T0801ABK.ZTC1LOG>
[file ] LOGFILERETENTION     <10>
[def  ] LOGFORMATCONSOLE     <93>
[def  ] LOGFORMATEMS         <16>
[def  ] LOGFORMATFILE        <93>
[file ] LOGLEVEL             <50>
[def  ] LOGLEVELCACHE        <50>
[def  ] LOGLEVELCONSOLE      <50>
[def  ] LOGLEVELEMS          <20>
[def  ] LOGLEVELFILE         <50>
[file ] LOGMAXFILELENGTH     <1000>
[def  ] MACS                 <hmac-sha1,hmac-md5,hmac-sha1-96,hmac-md5-96>
[file ] OWNER                <RoGeR>
[def  ] PARTIALSSHCOMACCESSGROUP1 <>
[def  ] PARTIALSSHCOMACCESSUSER1 <>
[def  ] PAUTHSUPPRESSIPADDRESS <FALSE>
[run  ] PORT                 <12229>
[def  ] PTCPIPFILTERKEY      <*>
[file ] PTYSERVER            <$ZPTYK>
[file ] RECORDDELIMITER      <ANY>
[def  ] RESTRICTIONCHECKFAILEDDEFAULT <FALSE>
[file ] SFTPALLOWGUARDIANCD  <FALSE>
[def  ] SFTPCPUSET           <>
[def  ] SFTPEDITLINEMODE     <none>
[def  ] SFTPEDITLINENUMBERDECIMALINCR <1000>
[def  ] SFTPEDITLINESTARTDECIMALINCR <-1>
[file ] SFTPENHANCEDERRORREPORTING <2>
[def  ] SFTPEXCLUSIONMODEREAD <SHARED>
[def  ] SFTPIDLETIMEOUT      <-1>
[def  ] SFTPMAXEXTENTS       <900>
[def  ] SFTPPRIMARYEXTENTSIZE <2>
[def  ] SFTPREALPATHFILEATTRIBUTEECHOED <FALSE>
[def  ] SFTPSECONDARYEXTENTSIZE <100>
[def  ] SFTPUPSHIFTGUARDIANFILENAMES <FALSE>
[def  ] SHELLENVIRONMENT     <>
[def  ] SOCKETKEEPALIVE      <1>
[def  ] SOCKETRCVBUF         <0>
[def  ] SOCKETSNDBUF         <0>
[def  ] SOCKTCPMAXRXMT       <0>
[def  ] SOCKTCPMINRXMT       <0>
[def  ] SOCKTCPRXMTCNT       <0>
[def  ] SOCKTCPTOTRXMTVAL    <0>
[def  ] SSHAUTOKEXBYTES      <1073741824>
[def  ] SSHAUTOKEXTIME       <3600>
[file ] SSHCTL               <SSHDBK>
```

```
[file ] SSHCTLAUDIT          <FALSE>
[def  ] SSHKEEPALIVETIME     <60>
[def  ] STOREDPASSWORDSONLY  <FALSE>
[file ] STRICTHOSTKEYCHECKING <FALSE>
[run  ] SUBNET               <$ZTC1>
[def  ] SUPPRESSCOMMENTINSSHVERSION <FALSE>
[def  ] TCPIPHOSTFILE        <*>
[def  ] TCPIPNODEFILE        <*>
[def  ] TCPIPRESOLVERNAME    <*>
$SSH01|20Jan14 15:34:01.55|10|CRYPTOPP version
T9999H06_12Dec2013_comForte_CRYPTOPP_0028
$SSH01|20Jan14 15:34:01.57|20|TCP/IP process is $ZTC1
$SSH01|20Jan14 15:34:05.35|20|Converted INTERFACE: 0.0.0.0
$SSH01|20Jan14 15:34:05.35|20|Converted INTERFACEOUT: 0.0.0.0
$SSH01|20Jan14 15:34:05.36|20|Define =TCPIP^PROCESS^NAME did not exist: Parameter
SUBNET was evaluated and define will be added.
$SSH01|20Jan14 15:34:05.36|20|DEFINE =TCPIP^PROCESS^NAME was set to <\BWNS02.$ZTC1>
$SSH01|20Jan14 15:34:05.37|20|SSH config database \BWNS02.$QAHPSSH.T0801ABK.SSHDBK is
not audited. A backup should be made after every config change.
$SSH01|20Jan14 15:34:05.39|10|SSH config database \BWNS02.$QAHPSSH.T0801ABK.SSHDBK
opened.
$SSH01|20Jan14 15:34:05.79|20|DEFINE =CIP^COMPAT^ERROR was set to
<\BWNS02.$QAHPSSH.T0801ABK.SUPPRESS>
$SSH01|20Jan14 15:34:05.79|20|DEFINE =SSH2^PROCESS^NAME was set to <\BWNS02.$SSH01>
$SSH01|20Jan14 15:34:05.80|10|Initializing SSH2 ADMIN run mode.
$SSH01|20Jan14 15:34:05.80|10|Initializing SSH2 CLIENT run mode.
$SSH01|20Jan14 15:34:05.80|10|Initializing SSH2 DAEMON run mode.
$SSH01|20Jan14 15:34:05.81|10|Loading private key from
\BWNS02.$QAHPSSH.T0801ABK.HOSTKEY
$SSH01|20Jan14 15:34:05.83|30|Host key algorithm: ssh-dss
$SSH01|20Jan14 15:34:05.84|30|Host key bits: 1024
$SSH01|20Jan14 15:34:05.84|30|Host key MD5 fingerprint:
26:ba:c4:e2:a7:1e:81:68:6c:18:10:49:96:50:04:03
$SSH01|20Jan14 15:34:05.84|30|Host key Bubble-Babble: xotam-patys-kupek-mogiv-tozul-
dihez-sevag-tikel-cebok-tityd-vyxux
$SSH01|20Jan14 15:34:05.86|10|SSH2 Server listening on process $ZTC1,
interface0.0.0.0, port 12229
```

Secure Shell Access to the NonStop™ Server

Note: This functionality is not enabled if you purchased a license restricted to file transfer ("HP NonStop SSH – SecureFTP" or "comForte SecurFTP/SSH").

SSH2 allows remote SSH clients to establish fully functional OSS shell sessions. SSH2 will also support the allocation of pseudo terminals (PTYs), which allow the remote users to execute full screen applications, such as vi or Emacs.

To Open an OSS Shell Using a Remote SSH Client

Note: This functionality requires OSS to be installed and running on your system.

After the STN and SSH2 processes have started successfully, you can now connect using an SSH client on a remote system. In the SSH command, you have to specify the Guardian userid and the IP address or host name that SSH2 is listening on:

```
m.horst@np-dev02:~> ssh comf.mh@10.0.0.199
The authenticity of host '10.0.0.199 (10.0.0.199)' can't be established.
DSA key fingerprint is 26:b8:77:fb:2f:22:81:3b:f6:44:4f:19:66:67:9a:be.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.199' (DSA) to the list of known hosts.
comf.mh@10.0.0.199's password:
$ ls
a1000       emsacstm    secret      t10mio      trace2      zrand1m
```

```
auditlog    ftps        sftpserv    taclcstm    tracecap    zz10mio
bashhist    fupcstm     shhistor    test        z1000000    zz1mio
bench       osstest     stna48      test101     z1mio       zzsa1894
benchcpu    osstzip     t1000       testbin     z1mio2      zzsa7884
benchs2k    rand1mio    t10000      testbin2    z1mio3      zzz10m
benchs3k    rs120157    t100000     testbin3    z1mioftp    zzz1mio
cryptand    scfcstm     t1000000    testbin4    z50mio
$
```

Please note that the Guardian userid is specified on the SSH command line.

Note: The very first time you connect, you will have to verify the authenticity of the host by adding the fingerprint of the host's public key to the trust list.

To Get a TACL Prompt Using a Remote SSH Client

You can also directly establish a connection to a TACL process, without involving any OSS functionality. Direct TACL access is provided by SSH2 as an SSH2 subsystem. You may connect to the TACL subsystem by specifying starting the remote SSH client with the –s option and "tacl" as subsystem name. Like with an ordinary shell session, you have to specify the Guardian userid and the IP address or host name, where SSH2 is listening on as parameters for the SSH command:

```
m.horst@np-dev02:~> ssh -s comf.mh@10.0.0.199 tacl
comf.mh@10.0.0.199's password:
TACL (T9205D46 - 19OCT2004), Operating System G06, Release G06.25.00
(C)1985 Tandem (C)2004 Hewlett-Packard Development Company, L.P.
CPU 1, process has no backup
February 10, 2006 13:09:41
(Invoking $SYSTEM.SYSTEM.TACLLOCL)
(Invoking $DATA1.MHHOME.TACLCSTM)
Current volume is $DATA1.MHHOME
1>
```

Note: Standard SSH clients will only support line mode interaction. You will not be able to invoke any block mode applications or applications that use advanced 6530 terminal features, unless using a SSH client supporting 6530 terminal sessions over SSH, such as comForte's MR-Win6530.

Secure Shell Access from NonStop™ to Remote Systems

Note: This functionality will be not be available with the SecurFTP/SSH and SecurTN products.

SSH2 includes two SSH clients, which allow the creation of secure shell sessions with a remote SSH daemon:

- SSHOSS is the OSS version of the SSH client. It provides fully functional terminal access to remote systems and, like SSH2 as a daemon, supports execution of full screen applications such as vi or Emacs, with the NonStop terminal as input and output device. It also allows establishing TCP and FTP port forwarding channels.
- SSH is the Guardian version of the SSH client. It allows you to create remote shells and execute remote commands and it supports port forwarding channels.

Note: SSH and SSHOSS will connect to a remote SSH daemon via a SSH2 process, which handles the SSH protocol layer.

To Connect to a Remote SSH Daemon with the NonStop SSH Client

You can create shell sessions with a remote SSH daemon both with the OSS SSH client (via SSHOSS) and the Guardian SSH client (via SSH).

From an OSS shell, run the SSHOSS client to create a secure shell session with a remote system as follows:

```
/home/mh: /G/data1/mhssh/sshoss comf.mh@10.0.0.201
SSH client version T9999H06_22Jan2014_comForte_SSHOSS_0097
WARNING: REMOTE HOST IDENTIFICATION UNKNOWN!
The host public key fingerprint is
  babble: xelol-vifez-cefis-gimiv-nepof-zemid-latut-zahoz-hyrun-hipop-hixex
  MD5:    04:bb:3c:a0:66:d4:bf:e3:60:b8:f3:31:49:d9:86:a6
Continue and add the host to the knownhost store(yes/no)? yes
Trying password authentication.
Enter m.horst@10.0.0.201's password:
Add password for m.horst@10.0.0.201 to the password store (yes/no)? no
Have a lot of fun...
m.horst@np-dev:~>
```

Note: For a production installation you may want to copy the SSHOSS program to an OSS standard bin directory, renaming it to "ssh". Alternatively, you may also create a symbolic link.

At the TACL prompt, run the SSH client to execute a command on a remote system as follows:

```
$DATA1 MHSSH 286> run ssh m.horst@10.0.0.201 whoami
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
You have no private keys in the key store.
Trying password authentication.
Enter m.horst@10.0.0.201's password:
Add password for m.horst@10.0.0.201 to the password store (yes/no)? no
m.horst
$SYSTEM ZSSH 287>
```

To Establish a Port Forwarding Tunnel with the NonStop SSH Client

Forwarding Local Port to Remote Port

You can create port forwarding channels for both the OSS SSH client (SSHOSS) and the Guardian SSH client (SSH). The following example illustrates how to establish a port forwarding tunnel for telnet sessions over SSH, using the Guardian SSH client:

```
$US SSH90 46> run ssh -N -L 5021:localhost:23 joe@10.0.0.111
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
```

The –N option suppresses the start of a remote shell. The –L option tells SSH2 to listen on port 5021 and forward any incoming connection to the remote SSH daemon and further to a telnet server on the same host, listening on port 23. The "localhost" in the command line refers to the target host of the forwarding tunnel, i.e. when using -L option this is the remote host.

After the SSH session is successfully established, the SSH process will wait until the SSH session is terminated or it is stopped. Thus, if you hit <break>, you can get the TACL prompt back and try to connect a telnet session over the SSH tunnel:

```
<break>
$US SSH90 47> telnet 127.0.0.1 5021
TELNET Client - T9558H01 - (19MAR12) - (IPMAAH)
Copyright Tandem Computers Incorporated 2004
Trying...Connected to 127.0.0.1.
Escape character is '^]'.

Welcome to SuSE Linux 8.2 (i586) - Kernel 2.4.20-4GB (0).

np-dev login:
```

In this example the local telnet client connects through the tunnel to the telnet server on remote host 10.0.0.111 that listens on loopback address 127.0.0.1, port 23.

Forwarding Remote Port to Local Port

Port forwarding channels can also be enabled in the opposite direction, i.e. from a remote port to a local port. The following example illustrates how to establish an SSH port forwarding tunnel from a remote host to the local host, using the Guardian SSH client:

```
$US SSH90A 48> run ssh -N -R 5021:localhost:23 testusr@10.0.0.234
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
```

The –N option suppresses the start of a remote shell. The –R option tells the remote SSH daemon on host 10.0.0.234 to listen on port 5021 and forward any incoming connection on that port to the local SSH2 process and this local process will further forward to a telnet server on the local host, listening on loopback address, port 23. The "localhost" in the command line refers to the target host of the forwarding tunnel, i.e. when using -R option this is the local host.

After the SSH session is successfully established, the SSH process will wait until the SSH session is terminated or it is stopped.

On the remote host 10.0.0.234 you can establish a telnet session over the SSH tunnel as follows:

```
testusr@linux-dev:~$ telnet 127.0.0.1 5021
TELNET Client - T9558H01 - (19MAR12) - (IPMAAH)
Copyright Tandem Computers Incorporated 1992-1997
Trying...Connected to 127.0.0.1.
Escape character is '^]'.


WELCOME TO npns01 [PORT $ZTC1 #23 WINDOW $ZTN0.#PTYSYNS]
TELSERV - T9553H01 - (25SEP2009) - (IPMAEP)


Available Services:

OSS      TACL     EXIT
Enter Choice>
```

In this example the remote telnet client started on host 10.0.0.234 connects through the tunnel to the telnet server on the local host that listens on loopback address 127.0.0.1, port 23.

Encrypted File Transfer

You can implement encrypted file transfers over SSH in various ways:

- Use the SFTP or SFTPOSS clients to initiate and control SFTP sessions from the NonStop server
- Use an SFTP client on a remote system to initiate and control SFTP sessions to the NonStop server from a remote system.
- Forward FTP connections over an SSH session.

To Connect a Remote SFTP Client to the NonStop Server

You can connect with an SFTP client on a remote system to SSH2 listening on the NonStop server as follows:

```
m.horst@np-dev02:~> sftp comf.mh@10.0.0.199
Connecting to 10.0.0.199...
comf.mh@10.0.0.199's password:
sftp> dir
a1000     auditlog  bashhist  bench     benchcpu  benchs2k  benchs3k  cryptand
emsacstm  ftps      fupcstm   osstest   osstzip   rand1mio  rs120157  scfcstm
secret    sftpserv  shhistor  ssh       stna48    t1000     t10000    t100000
t1000000  t10mio    taclcstm  test      test101   testbin   testbin2  testbin3
testbin4  trace2    tracecap  z1000000  z1mio     z1mio2    z1mio3    z1mioftp
z50mio    zrand1m   zz10mio   zz1mio    zzsa1894  zzsa7884  zzshgd    zzz10m
zzz1mio
sftp>
```

To Connect to a Remote SSH Daemon from the NonStop Server Using a NonStop SFTP Client

At the TACL prompt, run the SFTP client to create an SFTP session with a remote system as follows:

```
$DATA1 MHSSH 20> run sftp m.horst@10.0.0.201
SFTP client version T9999H06_22Jan2014_comForte_SFTP_0097
Connecting to 10.0.0.201...
You have no private keys in the key store.
Trying password authentication.
Enter m.horst@10.0.0.201's password:
Add password for m.horst@10.0.0.201 to the password store (yes/no)? no
sftp> ls -l
drwxr-xr-x    0 509      100           824 Jan 19 15:03 .
drwxr-xr-x    0 0        0             688 Nov 24 19:57 ..
-rw-r--r--    0 509      100          6340 Jun 19  2003 .Xdefaults
drwxr-xr-x    0 509      100           168 Jun 19  2003 Documents
-rw-r--r--    0 509      100        990000 Jan 19 15:00 ktest2
-rwxr-xr-x    0 509      100       1000000 Jan 19 14:58 ktestbig
drwxr-xr-x    0 509      100            80 Jun 19  2003 public_html
drwxr-xr-x    0 509      100           192 Nov 23 08:13 sshtest
sftp>
```

To Create an FTP Port Forwarding Tunnel with a NonStop SSH Client

You can establish FTP port forwarding channels for both the OSS SSH client (SSHOSS) and the Guardian SSH client (SSH). The following example illustrates this using the Guardian SSH client:

Run SSH as follows:

```
$DATA1 MHSSH 5> run ssh -N -L ftp/5021:localhost:21 m.horst@10.0.0.201
SFTP client version T9999H06_22Jan2014_comForte_SFTP_0097
You have no private keys in the key store.
Trying password authentication.
Enter m.horst@10.0.0.201's password:
Add password for m.horst@10.0.0.201 to the password store (yes/no)? no
```

The –N option suppresses the start of a remote shell. The -L ftp/5021:localhost:21 option tells SSH2 to listen on port 5021 and forward any incoming FTP connection to the remote SSH daemon and further to an FTP server on the same host, listening on port 21.

After the SSH session is successfully established, the SSH process will quietly wait until the SSH session is terminated or it is stopped. Thus, if you hit <break>, you can get the TACL prompt back and try to connect an FTP session over the SSH tunnel:

```
<break>
$DATA1 MHSSH 19> ftp
FTP Client - T9552J01 - (30MAR2012) - COPYRIGHT TANDEM COMPUTERS INCORPORATED 2012
ftp> open 127.0.0.1 5021
Connecting to 127.0.0.1.........Established.
220 np-dev.np-comforte.de FTP server (Version 6.5/OpenBSD, Linux port 0.3.3) rea
dy.
Name (127.0.0.1:user): m.horst
331 Password required for m.horst.
Password:
230- Have a lot of fun...
230 User m.horst logged in.
ftp> dir
200 PORT command successful.
150 Opening BINARY mode data connection for '/bin/ls'.
total 2062
-rw-r--r--  1 m.horst  users     6340 Jun 19  2003 .Xdefaults
drwxr-xr-x  5 m.horst  users      168 Jun 19  2003 Documents
-rw-r--r--  1 m.horst  users   990000 Jan 19 15:00 ktest2
-rwxr-xr-x  1 m.horst  users  1000000 Jan 19 14:58 ktestbig
drwxr-xr-x  2 m.horst  users       80 Jun 19  2003 public_html
drwxr-xr-x  3 m.horst  users      192 Nov 23 08:13 sshtest
226 Transfer complete.
1766 bytes received in  0.05 seconds (34.49 Kbytes/s)
ftp>
```

Due to the nature of the FTP protocol the forwarding of an FTP session is more complex than for example a telnet session (an FTP session usually consists of a data and a control channel, each established in a different direction). The remote SSH daemon must support the forwarding of FTP sessions (not all SSH daemon implementations are able to handle FTP forwarding).

Similar to the example under "Forwarding Remote Port to Local Port" in section "To Establish a Port Forwarding Tunnel with the NonStop SSH Client", the -R option can be used to forward an FTP connection from a remote host to the local host.

To Connect a Remote SCP Client to the NonStop Server

The SCPOSS object must be available in OSS name space under the name scp and must be found via the PATH environment variable. This can be achieved by creating a symbolic link to the installation location, e.g.

```
ln -s /G/system/zssh/scposs /usr/bin/scp
```

The environment variable ENV must be set via user attribute SHELL-ENVIRONMENT to ensure the PATH environment variable gets set appropriately. This can be achieved, e.g. by altering the user as follows (/etc/profile is just an example and often not a good choice):

```
ALTER USER test.us, SHELL-ENVIRONMENT /etc/profile
```

Ensure that shell scripts executed via ENV do not produce any output on stdout.

After the preparation is done you can connect with an SCP client on a remote system to SSH2 listening on the NonStop server as follows:

```
test@np-dev02:~/testsftp> rm bigtxt
test@np-dev02:~/testsftp> scp test.us@10.0.0.196:bigtxt .
test.us@10.0.0.196's password:
bigtxt                                        100%  640KB 640.0KB/s   00:00
test@np-dev02:~/testsftp> ls bigtxt
bigtxt
```

Using Public Keys to Authenticate Remote Users

This section describes how SSH2 can authenticate remote users using public keys. This involves creating a public key for the user on the remote system, and making the public key known to SSH2 on the NonStop server. After performing the steps described below, you should be able to connect to the NonStop server with your remote SSH or SFTP client using only the public key, without entering the NonStop user's password (you may still be prompted for the private key passphrase, though).

For additional information on public key authentication, please refer to the "Public Key Authentication" section in the "SSH Protocol Reference" chapter.

To Generate a Key Pair on an OpenSSH System

On the remote system, use the following command of OpenSSH (for details of key generation, please refer to the OpenSSH documentation):

```
>ssh-keygen -t dsa -C "comf.mh@10.0.0.199"
Generating public/private dsa key pair.
Enter file in which to save the key (/home/m.horst/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/m.horst/.ssh/id_dsa.
Your public key has been saved in /home/m.horst/.ssh/id_dsa.pub.
The key fingerprint is:
87:34:41:65:e5:df:e3:30:f6:46:22:02:19:24:1e:f2 comf.mh@10.0.0.199

>
```

Now the SFTP client will use this key whenever it connects to 10.0.0.199.

To Add the Public Key to the NonStop SSH2 User Database

Before a user can connect using public key authentication, the public key needs to be added to the user database. Using the SSHCOM component on the NonStop server, add the public key to the user as shown in the following example (note that the fingerprint was copied from the output of the previous step):

```
$DATA1 SSH2 12> sshcom $ssh01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 15:42:47.440
OPEN $ssh01
% ALTER USER comf.mh, publickey key1 fingerprint
87:34:41:65:e5:df:e3:30:f6:46:22:02:19:24:1e:f2, sftp-initial-directory /home/mh
OK, user comf.mh altered
% exit
exit
$DATA1 SSH2 13>
```

Note: The ALTER USER command will only work if the user already exists in the SSH2 userbase. This will be the case if you followed the other quick tour steps. You may also create a new user with the SSHCOM ADD USER command.

After this step you can now retry the step "To connect to a remote SSH daemon with the NonStop SSH client ". You will not be prompted for the NonStop user's password. Instead, SSH2 will authenticate the user with the public key configured for the remote user.

Using Public Keys to Logon to Remote Systems

This section explains the steps required to use public keys to authenticate to the remote system with a NonStop SSH or SFTP client. This involves generating a key pair for the NonStop user and configuring the public key on the remote system.

For additional information on public key authentication, please refer to the "Public Key Authentication" section in the "SSH Protocol Reference" chapter.

Note: The commands illustrated in the following steps will implicitly depend on the user issuing the commands. It is assumed all commands executed under the same user ID.

To Generate a Key Pair for a NonStop User

First, we will generate the key pair and store the private key in the SSH2 user database using SSHCOM from a TACL prompt:

```
$DATA1 SSH2 7> run sshcom $ssh01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 15:42:47.440
OPEN $ssh01
% mode client
mode client
OK, switched to client mode
% generate key test1, type rsa, comment "Thomas key"
generate key comf.tb:test1, type rsa, comment "Thomas key"
OK, key comf.tb:test1 successfully generated
%
```

Now the key has been generated and stored in the database. The next step will export that key and configure it on the remote system.

To Export the Public Key and Configure it on the Remote System

The following command within SSHCOM will export the public part of the key just generated and write it into a file:

```
$DATA1 SSH2 7> run sshcom $ssh01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 15:42:47.440
OPEN $ssh01
% export key comf.tb:test1, file $data1.tbtmp.tbkey, format openssh
export key comf.tb:test1, file $data1.tbtmp.tbkey, format openssh
```

```
OK, key comf.tb:test1 exported
%
```

Note: If you are executing SSHCOM as SUPER.SUPER, you will need to switch to CLIENT mode before exporting the key. Please issue following command before the EXPORT KEY command:
MODE CLIENT

The file $data1.tbtmp.tbkey now needs to be transferred to the remote system in BINARY mode. Note that the file contains only the public key and therefore contains no sensitive information.

The public key exported to the tbkey file can now be transferred to the remote system. The next step will configure the public key for the remote user.

To Configure the Public Key on the Remote System

The OpenSSH implementation keeps a directory called ".ssh" for each user. A file named "authorized keys" is located in the .ssh directory that contains the public key of each trusted key of a remote system. In order to add the public key contained in the file created in the prior step, the UNIX command "cat" can be used to add the content to the existing content in the file. The following commands are again executed on the remote system, this time using "normal user" logon credentials.

```
burgt@np-dev:~> pwd
/home/burgt
burgt@np-dev:~> cd .ssh
burgt@np-dev:~/.ssh> more pubkey
ssh-rsa
AAAAB3NzaC1yc2EAAAABEQAAAIEAkdR/ncHRVEJteOC1EMSkMgrrXpdcc6Lkejp7mcFKYNa0tMqP4eknTyFXUX
2jm1K7AKDh1Je52aqNJTBAIPIM
Bt+HboBKwjuZtb2+f1HG4LEA71NymoVcuABVyr1DvWPtpNzCNjaD0qdkR9yMlDZH/DCD/OqdneLJQ8B3RXbKl1
U= TB's RSA key
burgt@np-dev:~/.ssh> cat pubkey >> authorized_keys
burgt@np-dev:~/.ssh>
```

In the commands above

- The user's home directory is /home/burgt .
- The public key was transferred to the remote system under the location /home/burgt/.ssh/pubkey .
- The final command adds pubkey to authorized_keys. Please note the double '>>'; if you use only one '>', you will overwrite authorized_keys with the content of pubkey.

After this step you can now retry the step "To connect with a remote SSH client". You will not be prompted for the remote user's password. Instead, SSH2 will use the key pair configured for your NonStop user ID.

Configuring and Running SSH2

Configuration Overview

Administrators can specify configuration parameters of SSH2 processes through each of the following means:

- A configuration file
- PARAM commands
- Startup command line parameters

These different options enable system administrators to easily manage installations with multiple SSH2 processes, including those running on multiple TCP/IP processes and ports as well as in different modes. For example, several SSH2 processes that have identical SSH configurations can share the same configuration file, which streamlines administration. On the other hand, process-unique parameters, such as the port to listen on, can be specified on the command line.

On startup, SSH2 parses the sources of configuration parameter. A single parameter may be specified in multiple sources, e.g. in the configuration file and on the startup command line. In this case, SSH2 will process parameters with the following precedence (highest to lowest):

1. PARAM parameter
2. Parameter from configuration file 2 (CONFIG2)
3. Parameter from configuration file 1 (CONFIG)
4. Startup line parameter

This means that a parameter given in the configuration file will override the value given for the same parameter on the startup line. Likewise, a parameter value given as a PARAM command will override any value specified in the configuration file.

All SSH2 parameters can be specified in any of the configuration parameter sources, except in the following instances:

- The run mode of an SSH2 process is specified explicitly on the command line as the first startup line parameter. This parameter defines the general functionality the SSH2 process will provide. (See the "Starting SSH2" section for details.)
- The configuration file to be used as a parameter source can only be specified as a PARAM or startup line parameter, not in a configuration file.

It is important to note that parameter names are case insensitive, regardless of the manner in which way they are specified.

The Configuration File

Configuration files can be modified with a standard NonStop editor, such as TEDIT. The name of the file that a SSH2 process should use as the configuration source is passed to the program during startup. (See the "Starting SSH2" section for details.)

The file contains entries in the following form:

```
parameter-name  parameter-value
```

Like in the standard TCP/IP configuration files, any lines starting with a "#" character are interpreted as comments. Following is a sample configuration file for running SSH2 as a server that provides SFTP functionality:

```
# sample configuration file for a SSH2 server

#-----------------------------------------------------------------------------------
# general settings

# TCP/IP process the server runs on
SUBNET       $ZTC1

# port where SSH2 listens for incoming SSH connections
# we use the well-known SSH port
PORT        22

# file name of host key file
HOSTKEY     hostkey

# file name of user database file
SSHCTL    SSHCTL

#-----------------------------------------------------------------------------------
# log configuration
# set the level
LOGLEVEL 50
# enable console logging to $0
LOGCONSOLE $0
# additionally log to file
LOGFILE $data1.ssh2.ssh2log
```

PARAM Commands

The following PARAM command can be used to set SSH2 configuration parameters:

```
PARAM <parameter name> <parameter value>
```

If the parameter value contains one or more commas, it must be included in double quotes (see PARAM command in the NonStop™ "TACL Reference Manual" for use of comma as separator):

```
PARAM <parameter name> "<parameter value>"
```

All available SSH2 parameters can be specified using PARAM commands. But please be aware of the limitations described in the TACL Reference Manual: "TACL reserves 1024 bytes of internal storage for parameters and their values. The number and length of parameters in effect are limited by this storage area."

The following example demonstrates how to use a PARAM command to start an SSH2 server listening on $ZTC03, port 22:

```
> PARAM PORT 22

> PARAM SUBNET $ZTC03

> RUN SSH2/ NAME $SSH02 / SERVER
```

Startup Line Parameters

SSH2 configuration parameters can be passed on the startup line as follows:

```
<parameter name> <parameter value>; <parameter name> <parameter value>; ...
```

The following example demonstrates how to start multiple SSH2 instances that share the same SSHCONF configuration file listening on different subnets using the same port:

```
> PARAM CONFIG SSHCONF
> RUN SSH2 /NAME $SSH00, CPU 0, NOWAIT/ SERVER; SUBNET $ZTC0; PORT 22
> RUN SSH2 /NAME $SSH01, CPU 1, NOWAIT/ SERVER; SUBNET $ZTC1; PORT 22
> RUN SSH2 /NAME $SSH02, CPU 2, NOWAIT/ SERVER; SUBNET $ZTC2; PORT 22
> RUN SSH2 /NAME $SSH03, CPU 3, NOWAIT/ SERVER; SUBNET $ZTC3; PORT 22
```

For a complete description of the RUN SSH2 command, see the "Starting SSH2" section.

Starting SSH2

Note: The SSH2 process must be started and run under the SUPER.SUPER logon. When started using a different user ID, the process will issue a warning message and terminate.

You create a SSH2 process by issuing a TACL RUN command using the following syntax:

```
RUN SSH2 / runoptions / mode [ ; paramname paramvalue; ... ]
```

Following is a description of each aspect:

- *runoptions* are the standard Guardian RUN options, such as IN, CPU, or TERM.
- *mode* defines the "run mode" of the SSH2 process. The so-called run mode defines which functionality that instance will allow. The following run modes are defined:

DAEMON	runs a daemon process that provides the SFTP service to remote clients. No other functionality is provided.
DAEMON_ADMIN	combines the run modes DAEMON and ADMIN
CLIENT	runs a process that allows local SFTP clients to connect to the SSH2 process. No other functionality is provided.
CLIENT_ADMIN	combines the run modes CLIENT and ADMIN
ADMIN	runs a process that allows SSHCOM instances to connect to the SSH2 process and to configure the user database. No other functionality is provided.
NOADMIN	combines the run modes DAEMON and CLIENT
ALL	combines all run modes

  (SERVER can be used instead of DAEMON)

- *paramname paramvalue; ...* is a list of SSH2 configuration parameter settings as described in the previous section.

Note: When you start SSH2 in NOWAIT mode, make sure you have disabled logging to the home terminal. To do so, set the following PARAM:
PARAM LOGCONSOLE *

SSH2 Parameter Reference

This section describes all available SSH2 parameters in alphabetical order. Note that parameter names are case insensitive, regardless of the source in which they appear.

Some of the parameters are also valid for clients, please reference section "FILE I/O parameters for SFTP/SFTPOSS".

Parameter Overview

The following table lists all available SSH2 parameters and their meanings:

Parameter	Meaning
ALLOWEDAUTHENTICATIONS	Sets the list of allowed authentications for users automatically added to SSHCTL
ALLOWEDSUBSYSTEMS	Sets the list of allowed subsystems which globally restricts the users' settings of ALLOWED-SUBSYSTEMS attribute.
ALLOWFROZENSYSTEMUSER	Controls whether ssh users with a frozen Safeguard user configured as SYSTEM-USER are allowed to authenticate.
ALLOWINFOSSH2	Controls who is allowed to execute SSHCOM command INFO SSH2.
ALLOWPASSWORDSTORE	Controls whether users are allowed to use stored passwords for connections to remote SSH daemons.
ALLOWTCPFORWARDING	Allows global configuration of TCP port forwarding.
AUDITCONSOLE	Determines whether audit messages are written to the console.
AUDITEMS	Determines whether audit messages are written to EMS.
AUDITFILE	Determines whether audit messages are written to a file.
AUDITFILERETENTION	Controls audit file rollover.
AUDITFORMAT	Controls the format of the audit messages that are written.
AUDITFORMATCONSOLE	Controls the format of the audit messages that are written to the console.
AUDITFORMATEMS	Controls the format of the audit messages that are written to EMS.
AUDITFORMATFILE	Controls the format of the audit messages that are written to a file.
AUDITMAXFILELENGTH	Controls the maximum size of the audit file.
AUTOADDAUTHPRINCIPAL	Controls whether the PRINCIPAL should be automatically added.
AUTOADDSYSTEMUSERS	Controls whether remote users can log on via SSH using a Guardian user ID or alias, without configuring them explicitly via SSHCOM in the SSHCTL.
AUTOADDSYSTEMUSERSLIKE	Allows definition of a default user configuration when users are automatically added to SSHCTL.
BACKUPCPU	Specifies a backup CPU for running SSH2 as a NonStop process pair.
BANNER	Configures an authentication banner message to be displayed to SSH clients connecting to the SSH2 daemon.
BURSTSUPPRESSION	Controls log message duplicates suppression for all log targets.
BURSTSUPPRESSIONEXPIRATIONTIME	Configures the time interval duplicate log messages are suppressed before they get logged again.
BURSTSUPPRESSIONMAXLOGLEVEL	Sets the maximum log level of messages that get suppressed if burst suppression enabled.
CACHEBURSTSUPPRESSION	Controls log message duplicates suppression for log target memory cache.
CIPCOMPATERROR	Allows creation of DEFINE =CIP^COMPAT^ERROR.
CIPHERS	Details the list of cipher suites that will be accepted.

Parameter	Meaning
CLIENTALLOWEDAUTHENTICATIONS	Allows restriction of possible authentication methods used by NonStop ssh clients
CLIENTMODEOWNERPOLICY	Defines security granularity for client mode SSH2 database.
COMPRESSION	Specifies whether compressed SSH sessions are supported.
CONFIG	Specifies the file name of an SSH2 configuration file.
CONFIG2	Specifies the file name of a second configuration file for an SSH2 process.
CONSOLEBURSTSUPPRESSION	Controls log message duplicates suppression for log target console (home terminal).
CPUSET	Specifies the default value for USER attribute CPU-SET.
CUSTOMER	Allows setting the customer name or overwriting the customer name in the license file.
DAEMONMODEOWNERPOLICY	Defines security granularity for daemon mode USER records in the SSH2 database.
DISCONNECTIFUSERUNKNOWN	Controls the handling of unknown user names in incoming connections.
DNSMODE	Can be used to configure IP host name resolving regarding the use of multiple IP addresses per host name.
EMSBURSTSUPPRESSION	Controls log message duplicates suppression for log target EMS.
ENABLESTATISTICSATSTARTUP	Enables or disables statistics at startup.
FILEBURSTSUPPRESSION	Controls log message duplicates suppression for log target log file.
FULLSSHCOMACCESSGROUP<j>	Parameter set allows granting administrative SSHCOM command privileges to groups.
FULLSSHCOMACCESSUSER<i>	Parameter set allows granting administrative SSHCOM command privileges to users.
GSSAUTH	Enables or disables GSSAPI authentication.
GSSGEXKEX	Enables or disables GSSAPI key exchange with group exchange.
GSSKEX	Enables or disables GSSAPI key exchange.
GUARDIANATTRIBUTESEPARATOR	Specifies an additional separator character for Guardian file attributes.
HOSTKEY	Specifies the file name of host key file.
HOSTKEYBITS	Can be used to configure the size of of a newly generated local host key.
HOSTKEYTYPE	Can be used to select the type of a newly generated local host key.
INTERFACE	Specifies one or more local IP addresses or host names SSH2 should listen on for incoming SSH connections.
INTERFACEOUT	Specifies one or more local IP addresses or host names SSH2 should use for outgoing SSH connections.
INTERVALLIVEPRIVATEUSERKEY	Determines the period a newly generated user private key is in state 'LIVE' (before getting 'EXPIRED').
INTERVALLIVEPUBLICUSERKEY	This parameter is related to a user public key's life-cycle (configuration of database entity USER). It determines the length of the interval a user public key stays in state 'LIVE'.
INTERVALPENDINGPRIVATEUSERKEY	Determines the period a newly generated user private key is in state 'PENDING' (before getting 'LIVE').
INTERVALPENDINGPUBLICUSERKEY	This parameter is related to a user public key's life-cycle (configuration of database entity USER). It determines the length of the interval a user public key stays in state 'PENDING' after creation before it switches to state 'LIVE'.
IPMODE	Specifies IP mode of the SSH2 process
LICENSE	Specifies the location for the license file of SSH2.

Parameter	Meaning
LIFECYCLEPOLICYPRIVATEUSERKEY	Controls life cycle of user generated private keys.
LIFECYCLEPOLICYPUBLICUSERKEY	Controls the life-cycle of user public keys.
LOGCACHEDUMPONABORT	Determines if the internal log cache is written to the log file in case of process aborting.
LOGCACHESIZE	Determines the size of the internal log cache.
LOGCONSOLE	Determines whether log messages are written to a console.
LOGEMS	Determines whether log messages are written to EMS.
LOGEMSKEEPCOLLECTOROPENED	Controls opening/closing of the EMS collector.
LOGFILE	Determines whether log messages are written to a file.
LOGFILERETENTION	Controls log file rollover.
LOGFORMAT	Controls the format of the log messages that are written.
LOGFORMATCONSOLE	Controls the format of the log messages that are written to the console.
LOGFORMATEMS	Controls the format of the log messages that are written to EMS.
LOGFORMATFILE	Controls the format of the log messages that are written to a file.
LOGLEVEL	Sets the general logging level.
LOGLEVELCACHE	Determines whether log messages are written to the internal log cache.
LOGLEVELCONSOLE	Determines which messages will be written to the console.
LOGLEVELEMS	Determines which messages will be written to EMS.
LOGLEVELFILE	Determines which messages will be written to the log file.
LOGMAXFILELENGTH	Controls the maximum size of the log file.
LOGMEMORY	Allows regular logging of SSH2's memory usage to the log output.
MACS	Allows message authentication codes.
PARTIALSSHCOMACCESSGROUP<n>	Allows granting limited administrative SSHCOM command privileges to groups rather than just super.super.
PARTIALSSHCOMACCESSUSER<k>	Allows granting limited administrative SSHCOM command privileges to users other than super.super.
PAUTHSUPPRESSIPADDRESS	Can be used to suppress the IP address in USER_AUTHENTICATE_ calls.
PORT	The port the SSH2 server listens on for incoming connections.
PROPAGATEDEFINES	Controls whether SSH2 propagates defines in the SSH2 process context to newly started processes.
PTCPIPFILTERKEY	Sets the filter key to enable round-robin filtering.
PTCPIPFILTERTCPPORTS	Same effect as DEFINE =PTCPIP^FILTER^TCP^PORTS
PTYSERVER	Specifies the name of an STN process that functions as a pseudo terminal (PTY) server.
RECORDDELIMITER	Allows configuring the end-of-record marker used in binary file transfers into a structured NonStop file.
RESTRICTIONCHECKFAILEDDEFAULT	Allows all connection restriction checks to fail if a record for the Guardian user could not be found.
SAFEGUARD-PASSWORD-REQUIRED	Should be enabled if Safeguard is configured with PASSWORD-REQURED ON.
SFTPALLOWGUARDIANCD	Controls whether SFTPSERV allows a Guardian style CD command.
SFTPCPUSET	Specifies the default value for USER attribute SFTP-CPU-SET.
SFTPDISPLAYGUARDIAN	Controls output format (Guardian or OSS style) for SFTP informational messages.

Parameter	Meaning
SFTPEDITLINEMODE	Controls handling of Guardian edit lines that are longer than the maximum Guardian edit line length.
SFTPEDITLINENUMBERDECIMALINCR	Controls the Guardian edit line number decimal increment.
SFTPEDITLINESTARTDECIMALINCR	Defines at which line decimal incrementing of Guardian edit line numbers starts.
SFTPENHANCEDERRORREPORTING	Can be used to get more detailed file transfer error information.
SFTPEXCLUSIONMODEREAD	Defines file open exclusion mode of structured files.
SFTPIDLETIMEOUT	Controls whether SFTPSERV stops after specified user idle time.
SFTPMAXEXTENTS	Default value for MAXEXTENTS for files created on the NonStop system.
SFTPPRIMARYEXTENTSIZE	Default primary extend size for files created on the NonStop system.
SFTPREALPATHFILEATTRIBUTEECHOED	Helps using file attributes in SFTP commands with specific remote SFTP clients.
SFTPSECONDARYEXTENTSIZE	Default secondary extend size for files created on the NonStop system.
SFTPUPSHIFTGUARDIANFILENAMES	Defines that all Guardian file names are to be treated all upper or all lower case.
SHELLENVIRONMENT	Default value for USER attribute SHELL-ENVIRONMENT.
SOCKETKEEPALIVE	Specifies whether keep alive messages are enabled for TCP/IP sockets.
SOCKETRCVBUF	For setting the receive buffer size (socket option).
SOCKETSNDBUF	Allows setting the send buffer size (socket option).
SOCKTCPMAXRXMT	Allows setting maximum time for TCP retransmission timeout (socket option)
SOCKTCPMINRXMT	Allows setting minimum time for TCP retransmission timeout (socket option)
SOCKTCPRXMTCNT	Allows setting maximum number of continuous retransmissions prior to dropping a TCP connection (socket option)
SOCKTCPTOTRXMTVAL	Allows setting maximum continuous time spent retransmitting without receiving an acknowledgement from the other endpoint (socket option)
SSHAUTOKEXBYTES	Controls the frequency of key re-exchange on SSH sessions depending on the number of transferred bytes.
SSHAUTOKEXTIME	Controls the frequency of key re-exchange on SSH sessions depending on a timer.
SSHCTL	File name of user database.
SSHCTLAUDIT	Determines whether the user database file will be created as an audited file or not.
SSHKEEPALIVETIME	Controls the frequency of SSH "keepalive" messages.
STOREDPASSWORDSONLY	Disabling password prompt for authentication method password, allowing only to use stored passwords.
STRICTHOSTKEYCHECKING	Determines if local users are allowed to connect to unknown hosts.
SUBNET	Specifies one or more TCP/IP processes to use. DEFINE =TCPIP^PROCESS^NAME has precedence over this parameter.
SUPPRESSCOMMENTINSSHVERSION	Controls if SSH2 version is suppressed in the comment part of the ssh protocol version string exchanged between ssh client and ssh server
TCPIPHOSTFILE	Same effect as DEFINE =TCPIP^HOST^FILE.
TCPIPNODEFILE	Same effect as DEFINE =TCPIP^NODE^FILE.
TCPIPRESOLVERNAME	Same effect as DEFINE =TCPIP^RESOLVER^NAME.
USETEMPLATESYSTEMUSER	Allows using the same (dummy) Guardian user or *NONE* for automatically added users.

ALLOWEDAUTHENTICATIONS

Use this parameter to specify the authentication mechanisms that are allowed for system users that are automatically added to the SSHCTL database upon first login.

Parameter Syntax

```
ALLOWEDAUTHENTICATIONS (method[,method,...])
```

Arguments

method

Specifies an SSH authentication method to be allowed. Valid values are…

- password
  Password for the NonStop system's authentication mechanism. The password is validated against the SYSTEM-USER's password.
- publickey
  Public key authentication using the PUBLIC-KEYs configured for this user.
- keyboard-interactive
  Authentication according to RFC 4256 mapped to the standard GUARDIAN user authentication dialog verifying the SYSTEM-USER's password.
- gssapi-with-mic
  GSSAPI user authentication in accordance with the RFC 4462 standard. Including this method will also enable "gssapi-keyex" authentication, if the initial key exchange was performed over GSSAPI. See section "Single Sign-on with GSSAPI Authentication" for further details.

Default

If omitted, ALLOWEDAUTHENTICATIONS will be set to (keyboard-interactive,password,publickey).

Considerations

- ALLOWEDAUTHENTICATIONS is only relevant if AUTOADDSYSTEMUSERS is set to TRUE.
- ALLOWEDAUTHENTICATIONS will not override any list of authentication methods explicitly configured for a user (using SSHCOM ADD USER or ALTER USER).

Example

```
ALLOWEDAUTHENTICATIONS (keyboard-interactive,publickey)
```

See also

AUTOADDSYSTEMUSERS

ALLOWEDSUBSYSTEMS

This parameter can be used to globally restrict the SSH user settings to those subsystems listed in the value for ALLOWEDSUBSYSTEMS, which is a comma separated list of subsystem names. If a subsystem is not mentioned in both this global list and the SSH user's attribute ALLOWED-SUBSYSTEMS, then the incoming subsystem request will be denied.

Parameter Syntax

```
ALLOWEDSUBSYSTEMS subsystem[,subsystem,...]
```

Double quotes are required when setting the parameter via PARAM and more than one subsystem is listed:

```
PARAM ALLOWEDSUBSYSTEMS "sftp,tacl"
```

Arguments

subsystem

Specifies an SSH subsystem to be allowed for incoming connections. Valid values are…

- tacl
- sftp

Default

If omitted, ALLOWEDSUBSYSTEMS will be set to "sftp,tacl".

Considerations

- In an environment with more than one SSH2 process accessing the same SSHCTL database this parameter can be used to force users to use one SSH2 process for SFTP sessions and the other SSH2 process for TACL sessions.
- Although shell/exec requests are not subsystem requests, the parameter ALLOWEDSUBSYSTEMS can be used to generally prevent a user from starting a TACL: If parameter ALLOWEDSUBSYSTEM does not include subsystem tacl, then any request for a TACL is prevented even when ALLOW-CI is set to TRUE. If in this case CI-PROGRAM is configured as "*MENU* …" or "telnet …", i.e. a TACL is not directly started, then the telnet service menu or the telnet forwarding is processed as configured. A user cannot get a TACL prompt but it is possible to execute single commands in this case, see section "TACL Subsystem and Command Interpreter Configuration".

Example

```
ALLOWEDSUBSYSTEMS sftp
```

ALLOWFROZENSYSTEMUSER

This parameter controls the behavior when SSH2 detects that the configured SYSTEM-USER of the ssh user is in state FROZEN in Safeguard.

Parameter Syntax

```
ALLOWFROZENSYSTEMUSER TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether Safeguard users in state frozen are allowed to access the NonStop. Valid values are:

- TRUE: A frozen user is not rejected, i.e. can authenticate via configured authentication methods.
- FALSE: Authentication fails without trying any of the configured authentication methods if a Safeguard user is in state FROZEN.

Default

If omitted, ALLOWFROZENSYSTEMUSER will be set to FALSE. This is a change compared to releases prior to 0089 as frozen users were allowed before version 0089.

Considerations

- This parameter should be set to TRUE only if compatibility to previous behavior is required.
- Even if ALLOWFROZENSYSTEMUSER is set to TRUE, the methods password and keyboard-interactive will always fail due to the FROZEN state (because Safeguard is involved and will not authenticate a frozen user).

Example

```
ALLOWFROZENSYSTEMUSER FALSE
```

ALLOWINFOSSH2

This parameter defines the set of users that are allowed to execute the SSHCOM command INFO SSH2.

Parameter Syntax

```
ALLOWINFOSSH2 ALL|PARTIALSSHCOMACCESS|FULLSSHCOMACCESS
```

Arguments

ALL|PARTIALSSHCOMACCESS|FULLSSHCOMACCESS

Valid values are:

- ALL: Every user is allowed to execute SSHCOM command INFO SSH2.
- PARTIALSSHCOMACCESS: Only users configured with partial SSHCOM access are allowed to execute SSHCOM command INFO SSH2.
- FULLSSHCOMACCESS: Only users having full SSHCOM access are allowed to execute SSHCOM command INFO SSH2.

Default

If omitted, ALLOWINFOSSH2 will be set to ALL. This is compatible with the behavior before introduction of the parameter (i.e. prior to version 0092).

Example

```
ALLOWINFOSSH2 ALL
```

See also

FULLSSHCOMACCESSUSER<i>, FULLSSHCOMACCESSGROUP<j>, PARTIALSSHCOMACCESSUSER<k>, PARTIALSSHCOMACCESSGROUP<n>

ALLOWPASSWORDSTORE

This parameter controls whether users are allowed to use stored passwords for connections to remote SSH daemons.

Parameter Syntax

```
ALLOWPASSWORDSTORE TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether to allow password storage. Valid values are...

- TRUE: Any PASSWORDs stored for remote user ID will be automatically used for SSH password authentication. If no PASSWORD is stored for a connection, the user will be prompted after a successful authentication if a password should be stored in the password store.
- FALSE: Any stored PASSWORD will be ignored and users will not be prompted to interactively store passwords.

Default

If omitted, ALLOWPASSWORDSTORE will be set to TRUE.

Considerations

- If ALLOWPASSWORDSTORE is set to TRUE, passwords can be added manually to the user's password store using the SSHCOM ADD PASSWORD command.
  Passwords can also be added interactively, when users are prompted after a successful SSH password authentication with a remote SSH daemon.

Example

```
ALLOWPASSWORDSTORE TRUE
```

ALLOWTCPFORWARDING

Use this parameter to specify whether the SSH2 daemon will completely reject TCP port forwarding through SSH or allow TCP port forwarding depending on user configuration.

Parameter Syntax

`ALLOWTCPFORWARDING` *`TRUE|FALSE`*

Arguments

`TRUE|FALSE`

Specifies whether to allow port forwarding or not. Valid values are

- TRUE: port forwarding will be allowed unless user attribute ALLOW-TCP-FORWARDING is set to NO for a specific user.
- FALSE: port forwarding will be generally denied, independent of the value of user attribute ALLOW-TCP-FORWARDING.

Default

If omitted, SSH2 will reject port forwarding.

Considerations

This SSH2 parameter specifies on a global scope whether TCP port forwarding is allowed. Even if you set this parameter to TRUE, you may allow or deny port forwarding at the user level by setting the ALLOW-TCP-FORWARDING USER attribute. See the *SSHCOM Reference* for details.

Example

```
ALLOWTCPFORWARDING TRUE
```

AUDITCONSOLE

Use this parameter to define if and to what console device SSH2 audit messages are written to.

Parameter Syntax

`AUDITCONSOLE * | % | $0 |` *`auditdevice`*

Arguments

`*`

Signifies that no audit messages are written to a console.

`%`

Means that audit messages are written to the home terminal of the SSH2 process.

`$0`

Specifies that audit messages are written to $0.

`auditdevice`

Log messages are written the given device (e.g. $DEV.#SUBDEV).

Default

By default, no audit messages will be written ("*").

Considerations

- Although it is possible to specify a collector setting AUDITCONSOLE to a collector name is not recommended because a collector will cut long messages after 108 characters.
- If writing audit messages to a collector is required, then use parameter AUDITEMS instead.

See also

- AUDITEMS, AUDITFILE, AUDITFORMATCONSOLE
- "Audit Messages" in chapter "Monitoring and Auditing"

AUDITEMS

Use this parameter to define whether SSH2 audit messages are written to EMS.

Parameter Syntax

```
AUDITEMS collector | *
```

Arguments

*

Means that no audit messages are written to EMS.

collector

Specifies the name of the collector to which audit messages are written.

Default

By default, no audit messages are written to EMS ("*").

Considerations

- The AUDITFORMATEMS parameter controls the log message format.
- The parameter can be changed without having to restart SSH2, using the SSHCOM command interpreter (command SET AUDITEMS).
- To send audit messages to the default collector $0 use AUDITEMS $0.
- If the EMS collector specified cannot be opened during startup, SSH2 will write to the collector $0.
- If the EMS collector cannot be opened after it has been changed through SSHCOM, the original collector will stay active.

See also

AUDITFORMATEMS

AUDITFILE

Use this parameter to define whether SSH2 audit messages are written, and, if so, to what file.

Parameter Syntax

```
AUDITFILE * | filenameprefix
```

Arguments

`*`

Means that no audit log messages are written to a file.

`filenameprefix`

Specifies the prefix of the audit message file set. The actual audit file names are constructed from filenameprefix, which is appended by a number controlled by the AUDITFILERETENTION parameter.

Default

By default, no audit messages are written to a file ("*").

See also

- AUDITCONSOLE, AUDITFILERETENTION, AUDITFORMAT and AUDITMAXFILELENGTH
- "Audit Messages" in chapter "Monitoring and Auditing"

AUDITFILERETENTION

Use this parameter to control how many audit files SSH2 keeps when logfile rollover occurs.

Parameter Syntax

```
AUDITFILERETENTION n
```

Arguments

`n`

Specifies the number of audit files to keep.

Default

By default, 10 files are kept.

Considerations

- Setting the parameter to a value 0 disables log file retention.
- If log file retention is enabled, a minimum of 10 is enforced by this parameter.
- See section "Logfile/Auditfile Rollover" in the "Monitoring and Auditing" chapter for details on file rollover.
- The file security set for the current audit file (e.g. via FUP SECURE command) will be used for subsequently created audit files. The very first audit file will have the default file security of user SUPER.SUPER.

See also

AUDITMAXFILELENGTH and AUDITFILE

AUDITFORMAT

This parameter can be used to control the format of the audit messages that are written to the console and file. Set parameter AUDITFORMATCONSOLE and AUDITFORMATFILE to configure the audit format for console and file independently.

Parameter Syntax

```
AUDITFORMAT format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

bit 1 (decimal 1):	Date
bit 2 (decimal 2):	header (log messages a pre-fixed with "[log]")
bit 3 (decimal 4):	Time
bit 4 (decimal 8):	Milliseconds
bit 5 (decimal 16):	Process name
bit 7 (decimal 64)	Log level of message

Default

The default audit log format is 21 (date, time, process name).

Example

Display date, time, and milliseconds only:

```
AUDITFORMAT 13
```

Display date and time only:

```
AUDITFORMAT 5
```

See also

- AUDITCONSOLE, AUDITEMS, AUDITFILE, AUDITFORMATCONSOLE, AUDITFORMATEMS and AUDITFORMATFILE
- "Audit Messages" in the chapter entitled "Monitoring and Auditing"

AUDITFORMATCONSOLE

Use this parameter to control the format of the audit messages that are written to the console.

Parameter Syntax

```
AUDITFORMATCONSOLE format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

Bit 1 (decimal 1)	Date
Bit 2 (decimal 2)	Header (log messages a pre-fixed with "[log]")
Bit 3 (decimal 4)	Time
Bit 4 (decimal 8)	Milliseconds
Bit 5 (decimal 16)	Process ID (name or PIN)
Bit 7 (decimal 64)	Log level of message

Default

The default audit format is 21 (date, time, process name).

See also

- AUDITCONSOLE, AUDITFORMATEMS, AUDITFORMATFILE
- "Audit Messages" in the chapter entitled "Monitoring and Auditing"

AUDITFORMATEMS

Use this parameter to control the format of the audit messages that are written to EMS.

Parameter Syntax

```
AUDITFORMATEMS format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

Bit 1 (decimal 1)	Date
Bit 2 (decimal 2)	Header (log messages a pre-fixed with "[log]")
Bit 3 (decimal 4)	Time
Bit 4 (decimal 8)	Milliseconds
Bit 5 (decimal 16)	Process ID (name or PIN)
Bit 7 (decimal 64)	Log level of message

Default

The default audit format for EMS is 0 (none of the header fields).

See also

- AUDITEMS, AUDITFORMATCONSOLE, AUDITFORMATFILE
- "Audit Messages" in the chapter entitled "Monitoring and Auditing"

AUDITFORMATFILE

Use this parameter to control the format of the audit messages that are written to the log file.

Parameter Syntax

```
AUDITFORMATFILE format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

Bit 1 (decimal 1)	Date
Bit 2 (decimal 2)	Header (log messages a pre-fixed with "[log]")
Bit 3 (decimal 4)	Time
Bit 4 (decimal 8)	Milliseconds
Bit 5 (decimal 16)	Process ID (name or PIN)
Bit 7 (decimal 64)	Log level of message

Default

The default log format is 21 (date, time, process name).

See also

- AUDITFILE, AUDITFORMATCONSOLE, AUDITFORMATEMS
- "Audit Messages" in the chapter entitled "Monitoring and Auditing"

AUDITMAXFILELENGTH

Use this parameter to control the maximum size of an audit file.

Parameter Syntax

```
AUDITMAXFILELENGTH length
```

Arguments

length

A number representing the maximum log file length in kilobytes. Values must fall within the following constraints:

Maximum: 40.000 or 40 MB

Minimum: 100 KB

Default

The default length is 20000 KB.

Considerations

- Once a current audit file reaches the maximum size, a log rollover will occur. The current file will be closed and a new file will be opened. The new file will be named based on the audit round robin file set specified by the AUDITFILE and AUDITFILERETENTION parameters. If the file name already exists, any existing contents will be purged.

See also

- AUDITCONSOLE, AUDITFILE, AUDITFILERETENTION
- "Audit Messages" in the chapter titled "Monitoring and Auditing"

AUTOADDAUTHPRINCIPAL

Choose whether the PRINCIPAL should be automatically added if and only if either the 'password' or the 'keyboard-interactive' authentication method was successful and only if the 'gssapi-with-mic' authentication was executed successfully on Kerberos level but failed on SSH2 level only because none of the configured values for USER attribute PRINCIPAL matched the principal name found in the Kerberos ticket received from the SSH/SFTP/SCP client during authentication phase.

Parameter Syntax

```
AUTOADDAUTHPRINCIPAL TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether to add PRINCIPAL.

- TRUE: PRINCIPAL will be added if and only if either the 'password' or the 'keyboard-interactive' authentication method was successful and only if the 'gssapi-with-mic' authentication was executed successfully on Kerberos level.
- FALSE: PRINCIPAL will not be added even when either the 'password' or the 'keyboard-interactive' authentication method was successful and the 'gssapi-with-mic' authentication was executed successfully on Kerberos level.

Default

If omitted, AUTOADDAUTHPRINCIPAL is set to FALSE.

Example

```
AUTOADDAUTHPRINCIPAL TRUE
```

AUTOADDSYSTEMUSERS

Use this parameter to control whether remote users can log on via SSH using a Guardian user ID or alias, without configuring them explicitly via SSHCOM in the SSHCTL.

Parameter Syntax

AUTOADDSYSTEMUSERS *TRUE|FALSE*

Arguments

TRUE|FALSE

Specifies whether users logging on with a system User ID are automatically added to SSHCTL. Following are the two valid options:

- TRUE: system users are automatically added upon first login
- FALSE: logons of any user not contained in the SSHCTL will be denied.

Considerations

- Values of parameters AUTOADDSYSTEMUSERS, AUTOADDSYSTEMUSERSLIKE and USETEMPLATESYSTEMUSER are used together for automatic addition of SSH USER records:
  - If AUTOADDSYSTEMUSERS is FALSE, then the other two parameters will not be looked at, i.e. no SSH USER record added automatically.
  - If AUTOADDSYSTEMUSERS is TRUE and AUTOADDSYSTEMUSERSLIKE is not set, then parameter USETEMPLATESYSTEMUSER is not looked at. Assuming a client command like ssh <ssh-user>@host, the value of <ssh-user> is taken as SYSTEM-USER and a system user <ssh-user> must exist in order to successfully add the SSH USER entry automatically. All but SSH USER attributes user name and SYSTEM-USER are set to default values (ALLOWED-AUTHENTICATIONS attribute is taken from parameter ALLOWEDAUTHENTICATIONS if that is defined).
  - If AUTOADDSYSTEMUSERS is TRUE and AUTOADDSYSTEMUSERSLIKE is set, then parameter USETEMPLATESYSTEMUSER is checked: If parameter USETEMPLATESYSTEMUSER is FALSE, then the value of <ssh-user> is taken as SYSTEM-USER and a system user <ssh-user> must exist in order to successfully add the SSH USER entry automatically. All USER attributes but the SSH USER name and the SYSTEM-USER attribute are taken from the template user entry in this case. If parameter USETEMPLATESYSTEMUSER is TRUE, then all USER attributes but the SSH USER name, are taken from the template user entry, i.e. including the SYSTEM-USER attribute.

Default

If omitted, AUTOADDSYSTEMUSERS is set to FALSE.

Example

```
AUTOADDSYSTEMUSERS TRUE
```

See also

AUTOADDSYSTEMUSERSLIKE, USETEMPLATESYSTEMUSER

AUTOADDSYSTEMUSERSLIKE

Use this parameter to specify a user whose configuration in SSHCTL is used as default configuration when automatic adding of users to SSHCTL is enabled (i.e. if parameter AUTOADDSYSTEMUSERS has a value of TRUE).

Parameter Syntax

```
AUTOADDSYSTEMUSERSLIKE <user-name>
```

Arguments

<user-name>

The name of a user. The user must exist in the SSHCTL at the time a new user tries to logon and AUTOADDSYSTEMUSERS has a value of TRUE.

Considerations

- Any automatically added user will have the same attributes as the default user, except user name and system-user.
- In case the parameter AUTOADDSYSTEMUSERSLIKE is set to the name of a user not defined in SSHCTL and AUTOADDSYSTEMUSERS has a value of TRUE, then any authentication of a new user will be rejected.

Default

If omitted, a user is added with hard-coded default values if AUTOADDSYSTEMUSERS has a value of TRUE.

Example

```
AUTOADDSYSTEMUSERSLIKE comf.us
```

See also

AUTOADDSYSTEMUSERS, USETEMPLATESYSTEMUSER

BACKUPCPU

Use this parameter to run as a NonStop process pair.

Parameter Syntax

```
BACKUPCPU NONE|ANY|cpu
```

Arguments

NONE

SSH2 will not run as a process pair.

ANY

SSH2 will run as a NonStop process pair and will automatically select an available CPU for the backup process.

cpu

A number value that represents a CPU on your system. SSH2 will run as a NonStop process pair and will start the backup process in the specified CPU.

Considerations

To learn more about how SSH2 can help users leverage the fundamentals of the NonStop system to provide NonStop SSH access, please refer to the "NonStop Availability" section.

Default

If omitted, BACKUPCPU is set to NONE.

Example

```
BACKUPCPU ANY
```

BANNER

Use this parameter to configure an authentication banner message to be displayed to SSH clients connecting to the SSH2 daemon.

Parameter Syntax

`BANNER * | `*`filename`*

Arguments

`*`

Means no authentication banner is displayed.

`filename`

Specifies the file name containing the authentication banner to be displayed.

Considerations

- The BANNER file can be an edit file containing multiple lines.

Default

If omitted, BANNER is set to *.

Example

```
BANNER $SYSTEM.SSH2.BANNER
```

BURSTSUPPRESSION

Use this parameter to configure log burst suppression for log message duplicates of all log targets (EMS, console, file and memory cache).

Parameter Syntax

`BURSTSUPPRESSION `*`TRUE|FALSE`*

Arguments

`TRUE|FALSE`

Specifies whether BURSTSUPPRESSION is enabled or not:

- o TRUE: Duplicate log messages will be suppressed.
- o FALSE: Duplicate log messages will not be suppressed.

Considerations

When BURSTSUPPRESSION is TRUE, the log targets settings, enabled via target specific boolean parameters called CACHEBURSTSUPPRESSION, CONSOLEBURSTSUPPRESSION, EMSBURSTSUPPRESSION and FILEBURSTSUPPRESSION are ignored regardless of their value.

On the other hand when BURSTSUPPRESSION is FLASE the log targets settings, enabled via target specific boolean parameters called EMSBURSTSUPPRESSION, CONSOLEBURSTSUPPRESSION, FILEBURSTSUPPRESSION and CACHEBURSTSUPPRESSION are used.

When BURSTSUPPRESSION is TRUE and the BURSTSUPPRESSIONMAXLOGLEVEL is smaller than the log level assigned to a log message, then duplicates of that log message (targets of either cache, console, EMS or file) are not suppressed.

Default

If omitted, BURSTSUPPRESSION is set to FALSE.

Example

```
BURSTSUPPRESSION TRUE
```

See also

BURSTSUPPRESSIONEXPIRATIONTIME, BURSTSUPPRESSIONMAXLOGLEVEL, CACHEBURSTSUPPRESSION, CONSOLEBURSTSUPPRESSION, EMSBURSTSUPPRESSION and FILEBURSTSUPPRESSION

BURSTSUPPRESSIONEXPIRATIONTIME

Use this parameter to configure at what interval log burst suppression, for log messages of all log targets (EMS, Console, File and Cache), expires before a duplicate log messages is logged again.

Parameter Syntax

```
BURSTSUPPRESSIONEXPIRATIONTIME number-of-seconds
```

Arguments

number-of-seconds

Specifies the BURSTSUPPRESSIONEXPIRATIONTIME interval in seconds not to log duplicate log messages.

Considerations

BURSTSUPPRESSION or one of the log target specific parameters CACHEBURSTSUPPRESSION, CONSOLEBURSTSUPPRESSION, EMSBURSTSUPPRESSION and FILEBURSTSUPPRESSION need to be set to TRUE; otherwise the value of BURSTSUPPRESSIONEXPIRATIONTIME is ignored.

Default

If omitted, BURSTSUPPRESSIONEXPIRATIONTIME is set to 300.

Example

```
BURSTSUPPRESSIONEXPIRATIONTIME 240
```

See also

BURSTSUPPRESSION, BURSTSUPPRESSIONMAXLOGLEVEL

BURSTSUPPRESSIONMAXLOGLEVEL

Use this parameter to configure the maximum log level to suppress duplicate log messages for all log targets (EMS, console, file and cache).

Parameter Syntax

```
BURSTSUPPRESSIONMAXLOGLEVEL detail
```

Arguments

detail

A number is used to represent the level of suppression desired. A valid number must be between -1 indicating no suppression, and 100 indicating to suppress all duplicate log messages.

Considerations

Burst suppression (BURSTSUPPRESSION) is ignored for log messages with a log level greater than a maximum log level defined by parameter BURSTSUPPRESSIONMAXLOGLEVEL.

Default

If omitted, BURSTSUPPRESSIONMAXLOGLEVEL is set to 40.

Example

```
BURSTSUPPRESSIONMAXLOGLEVEL 50
```

See also

BURSTSUPPRESSION, BURSTSUPPRESSIONEXPIRATIONTIME

CACHEBURSTSUPPRESSION

Use this parameter to configure burst suppression for duplicate log message of log target memory cache.

Parameter Syntax

```
CACHEBURSTSUPPRESSION TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether CACHEBURSTSUPPRESSION is enabled or not:

- TRUE: Duplicate log messages will be suppressed.
- FALSE: Duplicate log messages will not be suppressed.

Considerations

The value of parameter CACHEBURSTSUPPRESSION is ignored if BURSTSUPPRESSION is set to TRUE.

Burst suppression for log target memory cache is enabled if either parameter BURSTSUPPRESSION or parameter CACHEBURSTSUPPRESSION is set to TRUE.

Default

If omitted, CACHEBURSTSUPPRESSION is set to FALSE.

Example

```
CACHEBURSTSUPPRESSION TRUE
```

See also

BURSTSUPPRESSION, BURSTSUPPRESSIONEXPIRATIONTIME, BURSTSUPPRESSIONMAXLOGLEVEL

CIPCOMPATERROR

In case there is no support for DEFINEs in the kernel (older OS releases), then a PARAM CIPCOMPATERROR can be set to SUPPRESS for a kernel process.

Parameter Syntax

```
CIPCOMPATERROR { SUPPRESS | * }
```

Arguments

SUPPRESS

DEFINE =CIP^COMPAT^ERROR will be set to SUPPRESS.

*

DEFINE =CIP^COMPAT^ERROR will not be set.

Default

The default for this parameter is *.

Considerations

Use this parameter to pass the value for the DEFINE =TCPIP^RESOLVER^NAME parameter to SSH2 servers configured as generic processes. This can also be achieved by adding the define =TCPIP^RESOLVER^NAME for the generic process (possible since G06.28/H06.06).

An existing DEFINE =CIP^COMPAT^ERROR passed to the SSH2 process at startup will remain in effect.

CIPHERS

Use this parameter to specify which cipher suites are admissible for the SSH2 server.

Parameter Syntax

```
CIPHERS suite [, suite, ...]
```

Arguments

suite

Specifies a cipher suite. Currently the following cipher suites are supported by SSH2:

- aes256-cbc: AES (Rijndael) in CBC mode, with 256-bit key
- aes128-cbc: AES with 128-bit key
- twofish256-cbc: Twofish in CBC mode, with 256-bit key
- twofish128-cbc: Twofish with 128-bit key
- twofish-cbc: alias for "twofish256-cbc" (Note: this is being retained for historical reasons)
- blowfish-cbc: Blowfish in CBC mode
- 3des-cbc: three-key 3DES in CBC mode
- arcfour: the ARCFOUR stream cipher
- cast128-cbc: CAST-128 in CBC mode

Considerations

For details about the ciphers listed above, please refer to standard SSH documentation, such as the manual for the RFCs available.

Default

If omitted, SSH2 will accept all ciphers mentioned above.

Example

```
CIPHERS 3des-cbc
```

This will enforce the use of only 3DES-encryption.

CLIENTALLOWEDAUTHENTICATIONS

Use this parameter to restrict the authentication methods the NonStop ssh clients (SSH[OSS], SFTP[OSS]) can try.

Parameter Syntax

```
CLIENTALLOWEDAUTHENTICATIONS method | "method,method,…"
```

Arguments

method

A supported authentication method

Considerations

- The value (list of authentication methods) is only relevant for outgoing ssh connections. For incoming connections the list of authentication methods is configured for each user (attribute ALLOWED-AUTHENTICATIONS).
- The authentication methods actually allowed at the client side consist of those methods that are specified in the client side option "AllowedAuthentications" as well as in the value of SSH2 parameter CLIENTALLOWEDAUTHENTICATIONS.

Default

The default value is to allow all methods that are supported.

Examples

```
CLIENTALLOWEDAUTHENTICATIONS "password,keyboard-interactive"
CLIENTALLOWEDAUTHENTICATIONS publickey
```

See also

- Ssh clients option AllowedAuthentications, see section "SSH and SFTP Client Reference", General Runtime options.
- User attribute ALLOWED-AUTHENTICATIONS

CLIENTMODEOWNERPOLICY

Defines security granularity for client mode SSH2 database.

Parameter Syntax

CLIENTMODEOWNERPOLICY *LOGINNAME* | *GUARDIANNAME* | *BOTH*

Arguments

LOGINNAME

The default owner is the login name, which can be a Guardian user identifier or an alias. An alias user cannot add/read/manipulate entries for the Guardian user the alias is configured with; vice versa, a Guardian user also can not add/read/manipulate entries for associated aliases. In other words, a Guardian or alias user can add/manipulate entries for that Guardian or alias user only.

The value LOGINNAME is recommended if different people are using the various aliases configured with the same Guardian user identifier.

GUARDIANNAME

The default owner is the Guardian user identifier, independent if the logon name is an alias or a Guardian user. Entries are read using the Guardian user ID only. This means that a Guardian user can add/read/manipulate entries for associated alias users, and vice versa.

The assumption is that the same person uses the aliases of a Guardian user identifier and the Guardian user identifier itself. This was the default before this enhancement was introduced (in release 89) and therefore value GUARDIANNAME needs to be used if the client mode policy of previous releases should be kept.

BOTH

The default owner is the login name but a guardian user can add or manipulate entries stored under an alias or a guardian user identifier. Entries are read for both the login name and the guardian user in case these are different (entries of the alias are read first, then entries of the guardian id). The value BOTH is only recommended if a guardian user and all aliases configured for this guardian user are solely used by one person and client mode records are to be stored under Guardian user identifier as well as alias names.

Example: Assume, an alias entry is present, but not an entry for the associated Guardian ID, and the user is logged on as the alias. With client mode owner policy set to LOGINNAME, privileges to read/alter the entry would be granted, for GUARDIANNAME they would not be granted because a matching entry is not found, and for BOTH they would be granted. If the Guardian entry is present but not the alias, and the user is logged on as the alias, LOGINNAME access would not be allowed, GUARDIANNAME would be allowed, and BOTH would also be allowed.

Considerations

- The value (list of authentication methods) is only relevant for outgoing ssh connections. For incoming connections the list of authentication methods is configured for each user (attribute ALLOWED-AUTHENTICATIONS).
- The authentication methods actually allowed at the client side consist of those methods that are specified in the client side option "AllowedAuthentications" as well as in the value of SSH2 parameter CLIENTALLOWEDAUTHENTICATIONS.

Default

The default value is BOTH.

Examples

```
CLIENTMODEOWNERPOLICY LOGINNAME
```

See also

Section on Ownership and Management of Client Mode Entities.

COMPRESSION

Use this parameter to specify whether compressed SSH sessions will be supported.

Parameter Syntax

`COMPRESSION` *`TRUE|FALSE`*

Arguments

TRUE|FALSE

The following arguments can be used to specify whether compression of the SSH session will be supported:

- TRUE: allows compressed sessions.
- FALSE: denies compressed sessions.

Default

If omitted, SSH2 will allow compressed sessions.

Example

```
COMPRESSION FALSE
```

CONFIG

Use this parameter to specify a configuration file for an SSH2 process.

Parameter Syntax

```
CONFIG file
```

Arguments

file

Specifies the name of the configuration file.

Default

If omitted, SSH2 will not use a configuration file.

Example

```
CONFIG $DATA1.SSH2.SSHCONF
```

Considerations

- This parameter can only be specified as PARAM or on the startup line. It is not valid within a configuration file.
- Parameters specified in the configuration file can be overwritten by PARAM or startup line settings.

CONFIG2

Use this parameter to specify a second configuration file for an SSH2 process.

Parameter Syntax

```
CONFIG2 * | cfgfile2
```

Arguments

*

Means no CONFIG2 file is used.

cfgfile2

Specifies the name of the second configuration file.

Default

If omitted, SSH2 will not use a second configuration file.

Example

```
CONFIG2 $DATA1.SSH2.SSHCONF2
```

Considerations

- The second configuration file has precedence over the first one.
- This parameter can only be specified as PARAM or on the startup line. It is not valid within a configuration file.
- Parameters specified in the configuration file can be overwritten by PARAM or startup line settings.

CONSOLEBURSTSUPPRESSION

Use this parameter to configure burst suppression for log message duplicates of log target console (home terminal).

Parameter Syntax

```
CONSOLEBURSTSUPPRESSION TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether CONSOLEBURSTSUPPRESSION is enabled or not:

- o TRUE: Duplicate log messages will be suppressed.
- o FALSE: Duplicate log messages will not be suppressed.

Considerations

Burst suppression (CONSOLEBURSTSUPPRESSION) is ignored if BURSTSUPPRESSION is set to TRUE.

Burst suppression for log target file is enabled if either parameter BURSTSUPPRESSION or parameter CONSOLEBURSTSUPPRESSION is set to TRUE.

Default

If omitted, CONSOLEBURSTSUPPRESSION is set to FALSE.

Example

```
CONSOLEBURSTSUPPRESSION TRUE
```

See also

BURSTSUPPRESSION, BURSTSUPPRESSIONEXPIRATIONTIME, BURSTSUPPRESSIONMAXLOGLEVEL

CPUSET

This parameter allows configuring the default set of CPUs the SSH2 process starts non-SFTPSERV user processes in.

Parameter Syntax

```
CPUSET cpu-set
```

Arguments

cpu-set

A comma separated list of CPU numbers or CPU number ranges defining allowed CPUs.

Default

If omitted, SSH2 will start all non-SFTPSERV processes in the CPU the SSH2 process is running in unless the USER record specifies a different CPU set for a specific user via attribute CPU-SET.

Example

```
CPUSET 2,4-6,9
```

Considerations

- A value configured in USER attribute CPU-SET has higher priority than the value defined in the SSH2 parameter CPUSET.
- CPU restrictions for processes dynamically started by STN can be established using option CPU of the ADD SERVICE STNCOM command. Please refer to the "STNCOM Commands" section for further details.

See also

SFTPCPUSET

CUSTOMER

Use this parameter to set the customer name or overwrite the customer name in the license file. If a customer name is set, either via license file or via parameter CUSTOMER, it will be used for encryption/decryption of the SSHCTL database records and the HOSTKEY file.

Parameter Syntax

```
CONFIG customer
```

Arguments

customer

Specifies the customer name. If spaces are included, then if the parameter value contains one or more commas or spaces, it must be included in double quotes.

Example

```
CUSTOMER "comForte 21 GmbH"
```

Considerations

- The parameter CUSTOMER has precedence over the customer name in the license file.
- When you plan to duplicate the host key and user database onto other NonStop systems (such as a disaster recovery system), you need to make sure the parameter CUSTOMER or the license file of that other system has the same customer name in it. Otherwise, the host key file and user data base cannot be used on the other system. If you purge the HOSTKEY and SSHCTL files and restart the SSH2 process, a new HOSTKEY and SSHCTL file will be created using either the value of parameter CUSTOMER or, if that does not exist, the customer name from the license file, if that exists.
- Although a license file is no longer required for NonStop SSH on H and J operating systems, any existing HOSTKEY and SSHCTL file requires the customer name that was used to create the file. If a license file exists, the customer name will be extracted from that file (entry SSH2.customer), unless parameter CUSTOMER is set in which case the value of CUSTOMER is used. If a license file does not exist and an existing HOSTKEY or SSHCTL file is accessed, the parameter CUSTOMER must be set to the original value for the customer name.
- For new installations without license file that include a creation of a new SSHCTL and HOSTKEY, there is no reason to set the CUSTOMER parameter.

See also

- HOSTKEY, SSHCTL

DAEMONMODEOWNERPOLICY

Defines security granularity for daemon mode USER records in the SSH2 database based on the OWNER field of the configured SSH user. Access to the daemon mode USER records in the SSH2 database will be granted in the same fashion as for PARTIALSSHCOMACCESSUSER/ PARTIALSSHCOMACCESSGROUP which is defined as partial access.

Access granted due to settings of FULLSSHCOMACCESSUSER/ FULLSSHCOMACCESSGROUP and PARTIALSSHCOMACCESSUSER/ PARTIALSSHCOMACCESSGROUP parameters and Safeguard OBJECTTYPE USER record are independent of the OWNER field. i.e. partial/full access granted via PARTIALSSHCOMACCESSUSER/ PARTIALSSHCOMACCESSGROUP and FULLSSHCOMACCESSUSER/ FULLSSHCOMACCESSGROUP parameters and Safeguard OBJECTTYPE USER record is not affected by this policy.

Parameter Syntax

```
DAEMONMODEOWNERPOLICY LOGINNAME | GUARDIANNAME | BOTH | NONE
```

Arguments

LOGINNAME

The login name value (which can be a guardian name or alias) of the guardian user that started the SSHCOM session will be compared to the OWNER field value (guardian name or alias) of the configured SSH user. This guardian user will have partial access to all the configured SSH user records and will be able to do SSHCOM INFO USER or SSHCOM ALTER USER commands on these records if a match was found using the login name value.

GUARDIANNAME

The guardian name of the login name value (which can be a guardian name or alias) of the guardian user that started the SSHCOM session will be compared to the OWNER field value (guardian name or alias) of the configured SSH user. This guardian user will have partial access to all the configured SSH user records and will be able to do SSHCOM INFO USER or SSHCOM ALTER USER commands on these records if a match was found using the guardian name of the login name value.

BOTH

The login name value (which can be a guardian name or alias) or guardian name of the login name value of the guardian user that started the SSHCOM session will be compared to the OWNER field value (guardian name or alias) of the configured SSH user. This guardian user will have partial access to all the configured SSH user records and will be able to do SSHCOM INFO USER or SSHCOM ALTER USER commands on these records if a match was found using the login name or guardian name of the login name values.

NONE

The OWNER field value of the configured SSH user will NOT be evaluated.

Considerations

- The DAEMONMODEOWNERPOLICY allows the same access rights to the daemon mode USER records as given by PARTIALSSHCOMACCESSUSER/ PARTIALSSHCOMACCESSGROUP.
- The DAEMONMODEOWNERPOLICY is only applicable when issuing SSHCOM INFO USER or SSHCOM ALTER USER commands in daemon mode.
- The logged in guardian user who started the SSHCOM session and is a group manager of the OWNER field value automatically has partial access rights to the daemon mode USER records.
- If DAEMONMODEOWNERPOLICY NONE was not specified, group managers, eg. <groupname>.manager, will always be treated as DAEMONMODEOWNERPOLICY BOTH regardless if LOGINNAME or GAURDIANNAME was specified.
- If SUPER.SUPER is denied full SSHCOM access via an OBJECTTYPE USER “DENY C” entry, the user SUPER.SUPER can still be configured as the owner of a USER record and would get partial access rights.

Also, SUPER.SUPER would have partial access rights for all USER records configured with a super group user as OWNER (if the policy is GUARDIANNAME or BOTH)

Default

The default value is NONE.

Examples

```
DAEMONMODEOWNERPOLICY LOGINNAME
```

See also

- FULLSSHCOMACCESSGROUP<j>, FULLSSHCOMACCESSUSER<i>, PARTIALSSHCOMACCESSGROUP<n> and PARTIALSSHCOMACCESSUSER<k>
- See "Security within SSHCOM" in section "SSHCOM Command Reference" about full and partial access rights.

DISCONNECTIFUSERUNKNOWN

Use this parameter to specify that incoming connections are immediately disconnected when the supplied SSH user name could not be found in the User Database.

Parameter Syntax

```
DISCONNECTIFUSERUNKNOWN TRUE|FALSE
```

Arguments

TRUE

The session will be disconnected immediately with indication "Access denied".

FALSE

A list of all supported authentication methods is sent back (this avoids returning the information that the user does not exist).

Default

The default for this parameter is FALSE.

Example

```
DISCONNECTIFUSERUNKNOWN TRUE
```

Considerations

- RFC 4252 allows both ways of processing requests of unknown users.
- If the parameter is not specified or is set to FALSE, the behavior is the same as before the parameter was introduced.

DNSMODE

When host names get resolved, multiple IP addresses may be the result for one host name. In versions before 0097 the first IP address of a possible list of IP addresses was always used. Starting with version 0097 the way how DNS name resolving is done regarding the use of multiple IP addresses per host name can be configured using parameter DNSMODE.

Parameter Syntax

```
DNSMODE FIRST|ALL
```

Arguments

FIRST|ALL

Specifies whether all IP addresses returned from a DNS server or only the first one are considered. Valid values are:

- FIRST for using just the first IP address.
- ALL for using all returned IP addresses.

Default

If omitted, FIRST is the default value, ensuring the DNS name resolving is handled as before introduction of this parameter.

Considerations

- One TCP/IP operation like listen or connect can only be done using exactly one IP address (which could be the ANY address in case of listen). See section "Multiple IP Process, Multiple IP Address Considerations" for more details.
- If DNS name resolving results in a list of IP addresses, then IPv4 and IPv6 IP addresses may appear in the list.
- The parameter setting is not only relevant for target host names specified by local SSH[OSS] and SSFTP[OSS] clients but also for names configured in parameter INTERFACE and INTERFACEOUT in that now multiple listens will be issued even if only one host name is configured for INTERFACE in case the DNS name resolving results in multiple IP addresses.
- Similarly, with DNSMODE ALL, local IP addresses used for outgoing connections are selected from a list of IP addresses in case multiple addresses are configured for a host name configured via INTERFACEOUT.

Example

```
DNSMODE ALL
```

See also

INTERFACE, INTERFACEOUT, IPMODE

EMSBURSTSUPPRESSION

Use this parameter to configure burst suppression for log message duplicates of log target of EMS.

Parameter Syntax

```
EMSBURSTSUPPRESSION TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether EMSBURSTSUPPRESSION is enabled or not:

- TRUE: Duplicate log messages will be suppressed.
- FALSE: Duplicate log messages will not be suppressed.

Considerations

The value of parameter EMSBURSTSUPPRESSION is ignored if BURSTSUPPRESSION is set to TRUE.

Burst suppression for log target EMS is enabled if either parameter BURSTSUPPRESSION or parameter EMSBURSTSUPPRESSION is set to TRUE.

Default

If omitted, EMSBURSTSUPPRESSION is set to FALSE.

Example

```
EMSBURSTSUPPRESSION TRUE
```

See also

BURSTSUPPRESSION, BURSTSUPPRESSIONEXPIRATIONTIME, BURSTSUPPRESSIONMAXLOGLEVEL

ENABLESTATISTICSATSTARTUP

This Boolean parameter allows enabling gathering statistics at startup of the SSH2 process.

Parameter Syntax

`ENABLESTATISTICSATSTARTUP` *`TRUE|FALSE`*

Arguments

`TRUE`

Statistics will be gathered immediately after the SSH2 process has started.

`FALSE`

Gathering statistical data will be enabled only after SSHCOM command ENABLE STATISTICS was issued.

Default

The default for this parameter is FALSE.

Example

```
ENABLESTATISTICSATSTARTUP TRUE
```

Considerations

- Maintaining statistics may slow down the SSH2 process.

FILEBURSTSUPPRESSION

Use this parameter to configure burst suppression for log message duplicates of log target of file.

Parameter Syntax

`FILEBURSTSUPPRESSION` *`TRUE|FALSE`*

Arguments

`TRUE|FALSE`

Specifies whether FILEBURSTSUPPRESSION is enabled or not:

- TRUE: Duplicate log messages will be suppressed.
- FALSE: Duplicate log messages will not be suppressed.

Considerations

The value of parameter FILEBURSTSUPPRESSION is ignored if BURSTSUPPRESSION is set to TRUE.

Burst suppression for log target file is enabled if either parameter BURSTSUPPRESSION or parameter FILEBURSTSUPPRESSION is set to TRUE.

Default

If omitted, FILEBURSTSUPPRESSION is set to FALSE.

Example

```
FILEBURSTSUPPRESSION TRUE
```

See also

BURSTSUPPRESSION, BURSTSUPPRESSIONEXPIRATIONTIME, BURSTSUPPRESSIONMAXLOGLEVEL

FULLSSHCOMACCESSGROUP<j>

This parameter set allows granting administrative SSHCOM command privileges to groups rather than just than super.super. Admin groups are defined via the parameter set FULLSSHCOMACCESSGROUP<j> where <j> is a number between 1 and 99.

Parameter Syntax

```
FULLSSHCOMACCESSGROUP<j> <group>
```

Arguments

`<group>`

A Guardian group name. All members of the group will have full SSHCOM access.

Default

By default, none of the parameters are set, i.e. only users configured in the Safeguard OBJECTTYPE USER record (if such exists) and super.super (unless explicitly denied in OBJECTTYPE USER) can access privileged commands.

Example

```
FULLSSHCOMACCESSGROUP1 admin
FULLSSHCOMACCESSGROUP2 super
```

Considerations

- Some of the privileged commands in SSHCOM are critical to the security of the system. Therefore granting access to other user accounts than super.super must be carefully considered.
- The parameters must be set contiguously, i.e. if one parameter FULLSSHCOMACCESSGROUP<k> is not defined the checking of FULLSSHCOMACCESSGROUP<i> parameters stops.
- This parameter set is disabled if a thawed OBJECTTYPE USER record exists in Safeguard, i.e. any FULLSSHCOMACCESSGROUP<j> parameter configuration is ignored in this case.

See also

- FULLSSHCOMACCESSUSER<i>
- See table in “SSHCOM Access Summary” in section "SSHCOM Command Reference".

FULLSSHCOMACCESSUSER<i>

This parameter set allows granting administrative SSHCOM command privileges to users other than super.super. Admin users are defined via the parameter set FULLSSHCOMACCESSUSER<i> where <i> is a number between 1 and 99.

Parameter Syntax

```
FULLSSHCOMACCESSUSER<i> <group>.<user>
```

Arguments

`<group>.<user>`

The Guardian logon name of the account that will have full SSHCOM access. Logon ids and alias names are not supported.

Default

By default, none of the parameters are set, i.e. only users configured in the Safeguard OBJECTTYPE USER record (if such exists) and super.super (unless explicitly denied in OBJECTTYPE USER) can access privileged commands.

Example

```
FULLSSHCOMACCESSUSER1 admin.joe
FULLSSHCOMACCESSUSER2 admin.jim
FULLSSHCOMACCESSUSER3 super.jane
```

Considerations

- Some of the privileged commands in SSHCOM are critical to the security of the system. Therefore granting access to other user accounts than super.super must be carefully considered.
- The user super.super has always full access to all SSHCOM commands unless explicitly denied in OBJECTTYPE USER record. Therefore it is not required to add super.super to the list of FULLSSHCOMACCESSUSER parameters.
- The parameters must be set contiguously, i.e. if one parameter FULLSSHCOMACCESSUSER<k> is not defined the checking of FULLSSHCOMACCESSUSER<i> parameters stops.
- This parameter set is disabled if a thawed OBJECTTYPE USER record exists in Safeguard, i.e. any FULLSSHCOMACCESSUSER<i> parameter configuration is ignored in this case.

See also

- FULLSSHCOMACCESSGROUP<j>
- See table in “SSHCOM Access Summary” in section "SSHCOM Command Reference".

GSSAUTH

Use this parameter to enable GSSAPI authentication in accordance with the RFC 4462.

Parameter Syntax

`GSSAUTH * | gssauth-process-name`

Arguments

`*`

GSSAPI user authentication is disabled

`Gssauth-process-name`

The process name of the GSSAUTH interface process that provides the GSSAPI functionality for SSH2.

Default

By default, GSSAPI authentication is disabled (*).

Example

```
GSSAUTH $GSS
```

Considerations

- The GSSAUTH interface process is part of the Kerberos installation on your NonStop Server.

See also

- GSSKEX, GSSGEXKEX, ALLOWEDAUTHENTICATIONS
- Section "Single Sign-on with GSSAPI Authentication".

GSSGEXKEX

Use this parameter to enable GSSAPI key exchange with group exchange, in accordance with the RFC 4462 standard (gss-gex-sha1-* key exchange algorithms).

Parameter Syntax

```
GSSGEXKEX TRUE|FALSE
```

Arguments

TRUE

GSSAPI kex with group exchange is enabled.

FALSE

GSSAPI kex with group exchange is disabled.

Default

By default, GSSAPI key exchange with group exchange is disabled (FALSE).

Considerations

- GSSGEXKEX is ignored if GSSAUTH is set to “*” (disabled) or GSSKEX is set to FALSE (disabled).
- Enabling GSSGEXKEX may cause problems with an SSH client if there is a faulty implementation of GSS key exchange with group exchange.

See also

- GSSAUTH, GSSKEX, ALLOWEDAUTHENTICATIONS
- Section "Single Sign-on with GSSAPI Authentication".

GSSKEX

Use this parameter to enable GSSAPI key exchange in accordance with RFC 4462.

Parameter Syntax

```
GSSKEX TRUE|FALSE
```

Arguments

TRUE

GSSAPI key exchange is enabled.

FALSE

GSSAPI key exchange is disabled.

Default

By default, GSSAPI key exchange is enabled (TRUE).

Considerations

- GSSKEX only takes effect if GSSAPI authentication is enabled. GSSKEX is ignored if GSSAUTH is set to “*” (disabled).

See also

- GSSAUTH, GSSGEXKEX, ALLOWEDAUTHENTICATIONS
- Section "Single Sign-on with GSSAPI Authentication".

GUARDIANATTRIBUTESEPARATOR

The value (which should only consist of one character) is used as additional separator character between Guardian file name and Guardian file attributes.

Use this parameter to specify additional separator character between Guardian file name and Guardian file attributes. The standard separator is always supported.

Parameter Syntax

`GUARDIANATTRIBUTESEPARATOR` *`separator`*

Arguments

`separator`

The character to be allowed as a separator of Guardian file attributes.

Considerations

- Use this parameter if a SFTP client does not support using commas in remote filenames.
- The configured separator character does not replace the default (which is comma) but is an alternate.
- Either the configured separator or the standard separator (comma) is supported but not a mix of both.

Default

If omitted, the only separator character is the comma.

Examples

```
GUARDIANATTRIBUTESEPARATOR -
GUARDIANATTRIBUTESEPARATOR "&"
```

HOSTKEY

Use this parameter to specify the filename of the host key file.

Parameter Syntax

`HOSTKEY` *`filename`*

Arguments

`filename`

Specifies the name of the host key file.

Considerations

- SSH2 generates the local host key during startup if the configured host key file does not exist. The type of the local host key is configurable via parameter HOSTKEYTYPE and the size of the key is determined by the value of parameter HOSTKEYBITS.

- The host key is the private key that is used to authenticate the host against the clients. The fingerprint of the host key will need to be configured on the remote systems that connect to the SSH2 process running on the NonStop system. The fingerprint of the host key file is displayed during startup of the process. It can also be seen via SSHCOM command INFO HOST-KEY.
- In order to prevent unauthorized usage of the host key file (i.e. moving it to other systems), the file is stored in a proprietary format and encrypted. The host key file is secured as "----".
- The customer name configured via parameter CUSTOMER or, if that does not exist, the customer name held within the license file for the SSH2 program is used as an input for host-based key encryption. When you plan to duplicate the host key and user database onto other NonStop systems (such as a disaster recovery system), you need to make sure the parameter CUSTOMER or the license file of that other system has the same customer name in it. Otherwise, the host key file and user data base cannot be used on the other system. If you purge the HOSTKEY and SSHCTL files and restart the SSH2 process, a new HOSTKEY and SSHCTL file will be created using either the value of parameter CUSTOMER or, if that does not exist, the customer name from the license file.
- Although a license file is no longer required for NonStop SSH on H and J operating systems, any existing HOSTKEY and SSHCTL file requires the customer name that was used to create the file. If a license file exists, the customer name will be extracted from that file (entry SSH2.customer), unless parameter CUSTOMER is set in which case the value of CUSTOMER is used. If a license file does not exist and an existing HOSTKEY or SSHCTL file is accessed, the parameter CUSTOMER must be set to the original value for the customer name.
- The public key part of the host key can be exported using the SSHCOM daemon mode command EXPORT HOST-KEY.
- If multiple SSH2 processes started from the same subvolume but used for different purposes, then not only separate SSH database files (configured via SSHCTL) but separate host key files (configured via HOSTKEY) should be configured. Example: SSH for maintenance and public network.

Default

If omitted, SSH2 will use a file name of HOSTKEY.

Example

```
HOSTKEY $SYSTEM.SSH2.SSHKEY
```

See also

CUSTOMER, HOSTKEYBITS, HOSTKEYTYPE

HOSTKEYBITS

A local host key is generated whenever the SSH2 process detects at startup that no local host key file exists. The size of local host key that gets generated can be configured using parameter HOSTKEYBITS.

Parameter Syntax

HOSTKEYBITS *keysize*

Arguments

keysize

Integer that specifies the size of the local host key in case one needs to be generated. Valid values are:

- o 1024 or 2048 if type of host key is RSA.
- o 1024 if type of host key is DSA.

Default

If omitted, 1024 is the default value, as before introduction of this parameter.

Considerations

- If a HOSTKEY file exists, then no new local host key is generated. In this case the value of parameter HOSTKEYBITS is not relevant.
- During startup, the key length of the local host key is now logged.
- In case a local host key is generated at startup of the SSH2 process, then the supported key size depends on the host key type: For type RSA key sizes 1024 and 2048 are supported, for type DSA only 1024 is supported.
- Key sizes 1024/2048 for RSA and 1024 for DSA have always been supported as remote host key sizes. The parameter HOSTKEYBITS is only relevant for local host keys.

Example

```
HOSTKEYBITS 2048
```

See also

HOSTKEY, HOSTKEYTYPE

HOSTKEYTYPE

A local host key is generated whenever the SSH2 process detects at startup that no local host key file exists. The type of the local host key that gets generated can be configured using parameter HOSTKEYTYPE.

Parameter Syntax

HOSTKEYTYPE *RSA|DSA*

Arguments

RSA|DSA

Specifies the type of the local host key in case one needs to be generated. Valid values are:

- RSA: The local host key will be of type RSA if newly generated at startup.
- DSA: The local host key will be of type DSA if newly generated at startup.

Default

If omitted, value DSA is the default value, as before introduction of this parameter.

Considerations

- If a HOSTKEY file exists, then no new local host key is generated. In this case the value of parameter HOSTKEYTYPE is not relevant.
- In case a local host key is generated at startup of the SSH2 process, then the supported key size depends on the host key type: For type RSA key sizes 1024 and 2048 are supported, for type DSA only 1024 is supported.
- Key types RSA and DSA have always been supported as remote host key types. The parameter HOSTKEYTYPE is only relevant for local host keys.

Example

```
HOSTKEYTYPE RSA
```

See also

HOSTKEY, HOSTKEYBITS

INTERFACE

Use this parameter to specify the local IP address(es) SSH2 should listen on for incoming SSH connections.

Parameter Syntax

```
INTERFACE ip-address [, ip-address, ...]
```

Arguments

ip-address

IP address or host name SSH2 should listen on.

Default

If omitted, SSH2 will listen on all local IP addresses of the configured TCPIP process(es) (SUBNET), which corresponds to INTERFACE value 0.0.0.0 or, in case of IPv6, 0::0.

Examples

```
INTERFACE 10.0.0.196

INTERFACE fe80::a00:8eff:fe00:d14e

INTERFACE ::FFFF:222.1.41.90

INTERFACE nonstop1
```

Considerations

- The value must be set consistent with the value of parameter IPMODE.
- If a host name is resolved to multiple IP addresses, then only those IP addresses are used that occur in the subnet configuration of the configured TCP/IP processes (parameter SUBNET).
- If the any address (0.0.0.0 or 0::0) is listed in INTERFACE, then the ANY address is used only for those IP processes that aren't configured with any of the other listed non-ANY addresses. See section "Multiple IP Process, Multiple IP Address Considerations" for an example.
- If parameter is set via PARAM and a comma separated list is defined, then the list must be enclosed in double quotes.

See also

DNSMODE, INTERFACEOUT, IPMODE, SUBNET

INTERFACEOUT

Use this parameter to specify the local IP address SSH2 should bind to for outgoing SSH connections.

Parameter Syntax

```
INTERFACEOUT ip-address [, ip-address, ...]
```

Arguments

ip-address

Local IP address or local host name SSH2 binds the TCP/IP socket to before connecting to a remote system.

Default

If omitted, SSH2 will bind to the IP address configured via parameter INTERFACE. If neither parameter INTERFACEOUT nor INTERFACE is set (or configured with value 0.0.0.0 / 0::0), any local IP addresses of the configured TCPIP process (SUBNET) will be used, selected by the TCPIP process.

Considerations

- The value must be set consistent with the value of parameter IPMODE.
- If a host name is resolved to multiple IP addresses, then only those IP addresses are used that occur in the subnet configuration of the configured TCP/IP processes (parameter SUBNET).
- If the any address (0.0.0.0 or 0::0) is listed in INTERFACEOUT, then the ANY address is used as bind address only for those IP processes that aren't configured with any of the other listed non-ANY addresses.
- If parameter is set via PARAM and a comma separated list is defined, then the list must be enclosed in double quotes.

Example

```
INTERFACEOUT 10.0.0.197
```

See also

DNSMODE, INTERFACE, IPMODE, SUBNET

INTERVALLIVEPRIVATEUSERKEY

This parameter is related to a user private key's life-cycle (configuration of database entity KEY). It determines the length of the interval a user private key stays in state 'LIVE'.

Parameter Syntax

`INTERVALLIVEPRIVATEUSERKEY number-of-`*`days`*

Arguments

`number-of-days`

The number of days a newly generated user private key will be in state 'LIVE' after leaving state 'PENDING' and before reaching state 'EXPIRED'.

Default

The default value for this parameter is 730, i.e. 2 years.

Example

```
INTERVALLIVEPRIVATEUSERKEY 1460
```

Considerations

- The life-cycle configuration of existing user private keys will not be modified due to this parameter. If existing keys need to participate in life-cycle control, then they must be configured via ALTER KEY command specifying the LIVE-DATE and EXPIRE-DATE command options.
- Parameter value is ignored if life-cycle for user private keys is disabled (i.e. if LIFECYCLEPOLICYPRIVATEUSERKEY is set to DISABLED).
- Parameter value is ignored if KEY attributes LIVE-DATE and EXPIRE-DATE are specified in GENERATE KEY and IMPORT KEY commands (if a user is allowed to specify these attributes according to the key life-cycle policy).

See also

LIFECYCLEPOLICYPRIVATEUSERKEY, INTERVALPENDINGPRIVATEUSERKEY

INTERVALLIVEPUBLICUSERKEY

This parameter is related to a user public key's life-cycle (configuration of database entity USER). It determines the length of the interval a user public key stays in state 'LIVE'.

Parameter Syntax

```
INTERVALLIVEPUBLICUSERKEY number-of-days
```

Arguments

`number-of-days`

The number of days a user public key will be in state 'LIVE' after leaving state 'PENDING' and before reaching state 'EXPIRED'.

Default

The default value for this parameter is 730, i.e. 2 years.

Example

```
INTERVALLIVEPUBLICUSERKEY 1460
```

Considerations

- The life-cycle configuration of existing user public keys will not be modified due to this parameter. If existing keys need to participate in life-cycle control, then they must be configured via ALTER USER, PUBLICKEY command specifying the LIVE-DATE and EXPIRE-DATE command options.
- Parameter value is ignored if life-cycle for user public keys is disabled (i.e. if LIFECYCLEPOLICYPUBLICUSERKEY is set to DISABLED).
- Parameter value is ignored if USER PUBLICKEY attributes LIVE-DATE and EXPIRE-DATE are specified in ALTER USER PUBLICKEY commands (if a user is allowed to specify these attributes according to the key lifecycle policy).

See also

LIFECYCLEPOLICYPUBLICUSERKEY, INTERVALPENDINGPUBLICUSERKEY

INTERVALPENDINGPRIVATEUSERKEY

This parameter is related to a user private key's life-cycle (configuration of database entity KEY). It determines the length of the interval a user private key stays in state 'PENDING' after creation before it switches to state 'LIVE'.

Parameter Syntax

```
INTERVALPENDINGPRIVATEUSERKEY number-of-days
```

Arguments

`number-of-days`

The number of days a newly generated user private key will be in state 'PENDING' after creation and before reaching state 'LIVE'.

Default

The default value for this parameter is 0, i.e. newly generated key will go into state 'LIVE' immediately if this parameter is not set to a different value than 0.

Example

```
INTERVALPENDINGPRIVATEUSERKEY 30
```

Considerations

- The life-cycle configuration of existing user private keys will not be modified due to this parameter. If existing keys need to participate in life-cycle control, then they must be configured via ALTER KEY command specifying the LIVE-DATE and EXPIRE-DATE command options.
- Parameter value is ignored if life-cycle for user private keys is disabled (i.e. if LIFECYCLEPOLICYPRIVATEUSERKEY is set to DISABLED).
- Parameter value is ignored if KEY attributes LIVE-DATE and EXPIRE-DATE are specified in GENERATE KEY and IMPORT KEY commands (if a user is allowed to specify these attributes according to the key life-cycle policy).

See also

LIFECYCLEPOLICYPRIVATEUSERKEY, INTERVALLIVEPRIVATEUSERKEY

INTERVALPENDINGPUBLICUSERKEY

This parameter is related to a user public key's life-cycle (configuration of database entity USER). It determines the length of the interval a user public key stays in state 'PENDING' after creation before it switches to state 'LIVE'.

Parameter Syntax

```
INTERVALPENDINGPUBLICUSERKEY number-of-days
```

Arguments

`number-of-days`

The number of days a user public key will be in state 'PENDING' after creation and before reaching state 'LIVE'.

Default

The default value for this parameter is 0, i.e. newly added user public keys will go into state 'LIVE' immediately if this parameter is not set to a different value than 0.

Example

```
INTERVALPENDINGPUBLICUSERKEY 30
```

Considerations

- The life-cycle configuration of existing user public keys will not be modified due to this parameter. If existing keys need to participate in life-cycle control, then they must be configured via ALTER USER PUBLICKEY command specifying the LIVE-DATE and EXPIRE-DATE command options.
- Parameter value is ignored if life-cycle for user public keys is disabled (i.e. if LIFECYCLEPOLICYPUBLICUSERKEY is set to DISABLED).
- Parameter value is ignored if USER PUBLICKEY attributes LIVE-DATE and EXPIRE-DATE are specified in ALTER USER PUBLICKEY commands (if a user is allowed to specify these attributes according to the key lifecycle policy).

See also

LIFECYCLEPOLICYPUBLICUSERKEY, INTERVALLIVEPUBLICUSERKEY

IPMODE

This parameter is used to set the IP mode the SSH2 process is running in. Depending on this parameter the SSH2 process supports IPv4 only, IPv6 only, or both.

Parameter Syntax

```
IPMODE ip-mode
```

Arguments

ip-mode

The IP mode the SSH2 process will be running in. The following IP modes are supported:

- IPV4 – TCP/IP version 4 is supported only
- IPV6 – TCP/IP version 6 is supported only
- DUAL – Both TCP/IP versions 4 and 6 are supported

Default

The default value for this parameter is IPV4.

Example

```
IPMODE IPv6
```

Considerations

- The IPMODE parameter of SSH2 corresponds to the TCP/IP monitor process option FAMILY. The configuration of SSH2 parameter SUBNET or define =TCPIP^PROCESS^NAME must not contradict the value of IPMODE, i.e. if IPMODE is set to IPv4, then the TCP/IP process cannot be configured with FAMILY IPv6 and vice versa.
- Similarly, the configuration of SSH2 parameters INTERFACE and INTERFACEOUT must be set consistently with setting of parameter IPMODE.

See also

SUBNET, INTERFACE, INTERFACEOUT

LICENSE

Use this parameter to specify a different location for the SSH2 license file.

Note: If you purchased NonStop SSH with the NonStop™ Operating System Kernel for H Series and J Series NonStop™ platforms, you will not need a license file anymore.

Parameter Syntax

```
LICENSE file
```

Arguments

file

Specifies the name of the SSH2 license file.

Considerations

- If the file name is not fully qualified, SSH2 will add the home subvolume of the object file to the file name.
- A license is no longer required for TNS/E systems. If a license file exists, then the customer name will be extracted from it.
- Please see the section on the HOSTKEY parameter for more information on the interaction of the license file with the host key file.

- Please see the section on the SSHCTL parameter for more information on the interaction of the license file with the SSH2 database.

Default

If omitted, an SSH2 process will search for a file named "LICENSE" on the subvolume where the SSH2 object resides.

LIFECYCLEPOLICYPRIVATEUSERKEY

This parameter controls the life-cycle of user generated private keys. If enabled, a ‘not valid before date’ and a ‘not valid after date’ can be defined for each individual key. This can be achieved by setting the dates explicitly via entity KEY attributes LIVE-DATE and EXPIRE-DATE or implicitly via globally defined length of the key pending time period after key generation and length of the period a key is in ‘LIVE’ state. Only a key in ‘LIVE’ state may be part of a publickey authentication of the user owning a private key.

Parameter Syntax

```
LIFECYCLEPOLICYPRIVATEUSERKEY DISABLED|FIXED|VARIABLE
```

Arguments

DISABLED

Life-cycle control for user generated private keys will not be enabled. When a key is generated it is immediately in state ‘LIVE’ and it will never expire.

FIXED

Users without full SSHCOM access cannot set or alter KEY attributes LIVE-DATE and EXPIRE-DATE. Both dates will be determined by the CREATION-DATE and the values of parameters INTERVALPENDINGPRIVATEUSERKEY and INTERVALLIVEPRIVATEUSERKEY.

VARIABLE

A user can specify the LIVE-DATE and EXPIRE-DATE when generating or importing a private key or when altering the private key. By not specifying these attributes in a GENERATE KEY or IMPORT KEY command, the values for LIVE-DATE and EXPIRE-DATE will be automatically set depending on the CREATION-DATE and the values of parameters INTERVALPENDINGPRIVATEUSERKEY and INTERVALLIVEPRIVATEUSERKEY.

Default

The default for this parameter is DISABLED resulting in the same behavior as before the introduction of this parameter.

Example

```
LIFECYCLEPOLICYPRIVATEUSERKEY FIXED
```

Considerations

- Users with full SSHCOM access can set or modify KEY attributes LIVE-DATE and EXPIRE-DATE even when the life-cycle policy for user private keys is set to FIXED.

See also

INTERVALLIVEPRIVATEUSERKEY, INTERVALPENDINGPRIVATEUSERKEY

LIFECYCLEPOLICYPUBLICUSERKEY

This parameter controls the life-cycle of user public keys. If enabled, a ‘not valid before date’ and a ‘not valid after date’ can be defined for each individual key. This can be achieved by setting the dates explicitly via entity USER PUBLICKEY attributes LIVE-DATE and EXPIRE-DATE or implicitly via globally defined length of the key pending time period after key addition and length of the period a key is in ‘LIVE’ state. Only a key in ‘LIVE’ state may be part of a public key authentication of the user configured with the key.

Parameter Syntax

```
LIFECYCLEPOLICYPUBLICUSERKEY DISABLED|FIXED|VARIABLE
```

Arguments

DISABLED

Life-cycle control for user public keys will not be enabled. When a public key is added, it is immediately in state ‘LIVE’ and it will never expire.

FIXED

Users without full SSHCOM access cannot set or alter KEY attributes LIVE-DATE and EXPIRE-DATE. Both dates will be determined by the CREATION-DATE and the values of parameters INTERVALPENDINGPUBLICUSERKEY and INTERVALLIVEPUBLICUSERKEY.

VARIABLE

Users with partial access can specify the LIVE-DATE and EXPIRE-DATE when adding a user public key or when altering the public key. By not specifying these attributes in an ALTER USER PUBLICKEY command, the values for LIVE-DATE and EXPIRE-DATE will be automatically set depending on the CREATION-DATE and the values of parameters INTERVALPENDINGPUBLICUSERKEY and INTERVALLIVEPUBLICUSERKEY.

Default

The default for this parameter is DISABLED resulting in the same behavior as before the introduction of this parameter.

Example

```
LIFECYCLEPOLICYPUBLICUSERKEY FIXED
```

Considerations

- Users with full SSHCOM access can set or modify USER PUBLICKEY attributes LIVE-DATE and EXPIRE-DATE even when the life-cycle policy for user public keys is set to FIXED.

See also

INTERVALLIVEPUBLICUSERKEY, INTERVALPENDINGPUBLICUSERKEY, FULLSSHCOMACCESSUSER<i>, FULLSSHCOMACCESSGROUP<j>, PARTIALSSHCOMACCESSUSER<k> and PARTIALSSHCOMACCESSGROUP<n>

LOGCACHEDUMPONABORT

Use this parameter to define whether SSH2 writes the log messages held in the log cache are written to the log file in case of an abort.

Parameter Syntax

```
LOGCACHEDUMPONABORT TRUE|FALSE
```

Arguments

TRUE

In case of abort the content of the log cache will be written to the configured log file.

FALSE

The content of the log cache will be discarded on process abort.

Default

The default for this parameter is TRUE.

Considerations

- The log cache content can be written to the log file at any time via SSHCOM command FLUSH LOGCACHE.

See also

- LOGCACHESIZE, LOGLEVELCACHE, LOGFILE
- "Log Messages" in the "Monitoring and Auditing" chapter.
- Commands FLUSH LOGCACHE and CLEAR LOGCACHE in the "SSHCOM Command Reference" chapter.

LOGCACHESIZE

Use this parameter to define how many lines of log messages are held in log cache.

Parameter Syntax

```
LOGCACHESIZE <lines>
```

Argument

`<lines>`

The number of log messages (lines) to be held in the log cache. The minimum value is 1024 and the maximum value is 1048576 (1024 * 1024).

Considerations

- The LOGLEVELCACHE parameter controls what messages are written to the log cache.

Default

By default, the minimum value (1024) is used.

See also

- LOGLEVELCACHE
- Commands SET LOGCACHESIZE in the "SSHCOM Command Reference" chapter.

LOGCONSOLE

Use this parameter to define whether SSH2 log messages are written to a console device, and, if so, which device.

Parameter Syntax

```
LOGCONSOLE * | % | $0 | logdevice
```

Arguments

`*`

Means that no log messages are written to a console device.

`%`

Results in log messages being written to the home terminal of the SSH2 process.

`$0`

Specifies that log messages are written to $0.

`logdevice`

Specifies that log messages are written to a given device (e.g. $DEV.#SUBDEV).

Considerations

- The LOGLEVELCONSOLE parameter controls what messages are produced by SSH2.
- Log messages are automatically cut by the collector when using value $0 for LOGCONSOLE. Please use LOGEMS to enable logging to an EMS collector.

Default

By default, log messages are written to the home terminal ("%").

See also

- LOGEMS, LOGFILE, LOGLEVELCONSOLE
- "Log Messages" in the "Monitoring and Auditing" chapter.

LOGEMS

Use this parameter to define whether SSH2 log messages are written to EMS.

Parameter Syntax

```
LOGEMS collector | *
```

Arguments

`*`

Means that no log messages are written to EMS.

`collector`

Specifies the name of the collector to which log messages are written.

Default

By default, no log messages are written to EMS ("*").

Considerations

- The LOGLEVELEMS parameter controls what messages are produced by SSH2.
- The LOGFORMATEMS parameter controls the log message format.
- The parameter can be changed without having to restart SSH2, using the SSHCOM command interpreter.
- To send messages to the default collector $0 use LOGEMS $0.
- If the EMS collector specified cannot be opened during startup, SSH2 will write to the collector $0.
- If the EMS collector cannot be opened after it has been changed through SSHCOM, the original collector will stay active.

See also

LOGLEVELEMS, LOGFORMATEMS

LOGEMSKEEPCOLLECTOROPENED

This Boolean parameter controls if the configured EMS collector (see LOGEMS) will be opened and closed for every log message.

Parameter Syntax

```
LOGEMSKEEPCOLLECTOROPENED TRUE|FALSE
```

Arguments

TRUE

The EMS collector will be opened once (and re-opened after errors only)

FALSE

The EMS collector will be opened and closed for each log message written to the EMS collector (configured via parameter LOGEMS)

Default

The default for this parameter is TRUE.

Example

```
LOGEMSKEEPCOLLECTOROPENED TRUE
```

Considerations

- Keeping the EMS collector open instead of opening and closing it for every log message will reduce overhead.
- Closing the collector for every log message is only required if the collector's supported maximum number of event message issuers is reached.

LOGFILE

Use this parameter to define whether SSH2 log messages are written, and, if so, to which file.

Parameter Syntax

```
LOGFILE * | file
```

Arguments

*

Means that no log messages are written to a file.

filenameprefix

Specifies the prefix of the log file set. The actual audit file names are constructed based on the prefix assigned and by a number generated based on the settings of the LOGFILERETENTION parameter.

Default

By default, no log messages are written to a file ("*").

Considerations

- The LOGLEVELFILE parameter controls what messages are produced by SSH2.
- The LOGFORMATFILE parameter controls the log message format.

See also

- LOGCONSOLE, LOGLEVELFILE, LOGFORMATFILE, LOGMAXFILELENGTH, LOGFILERETENTION
- "Log Messages" in the chapter entitled "Monitoring and Auditing".

LOGFILERETENTION

Use this parameter to control how many log files SSH2 keeps when log file rollover occurs.

Parameter Syntax

```
LOGFILERETENTION n
```

Arguments

n

Specifies the number of log files to keep.

Default

By default, 10 files are kept.

Considerations

- Setting the parameter to a value 0 disables log file retention.
- If log file retention is enabled, a minimum of 10 is enforced by this parameter.
- See section "Logfile/Auditfile Rollover" in the "Monitoring and Auditing" chapter for details on file rollover.
- The file security set for the current log file (e.g. via FUP SECURE command) will be used for subsequently created log files. The very first log file will have the default file security of user super.super.

See also

LOGMAXFILELENGTH, LOGFILE

LOGFORMAT

Use this parameter to control the format of the log messages that are written to the console or log file.

Parameter Syntax

```
LOGFORMAT format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

bit 1 (decimal 1):	Date
bit 2 (decimal 2):	Header (log messages a pre-fixed with "[log]")
bit 3 (decimal 4):	Time
bit 4 (decimal 8):	Milliseconds
bit 5 (decimal 16):	Process name
Bit 7 (decimal 64)	Log level of message

Default

The default log format is 93 (process name, date, time, milliseconds, and log level).

Example

Display date, time, and milliseconds only:

```
LOGFORMAT 13
```

Display date and time only:

```
LOGFORMAT 5
```

Considerations

- This parameter is retained for downward compatibility only and has been replaced by the parameters LOGFORMATCONSOLE and LOGFORMATFILE.
- If no value is set for the parameters LOGFORMATCONSOLE or LOGFORMATFILE, they will inherit their value from the parameter LOGFORMAT.
- If both LOGFORMATCONSOLE and LOGFORMATFILE are set with a value, the parameter of LOGFORMAT becomes meaningless.

See also

LOGFORMATCONSOLE, LOGFORMATEMS, LOGFORMATFILE

LOGFORMATCONSOLE

Use this parameter to control the format of the log messages that are written to the console.

Parameter Syntax

```
LOGFORMATCONSOLE format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

Bit 1 (decimal 1)	Date
Bit 2 (decimal 2)	Header (log messages a pre-fixed with "[log]")
Bit 3 (decimal 4)	Time
Bit 4 (decimal 8)	Milliseconds
Bit 5 (decimal 16)	Process ID (name or PIN)
Bit 7 (decimal 64)	Log level of message

Default

The default log format is 93 (date, time, milliseconds, process ID, and log level).

Example

Display date, time, and milliseconds only:

```
LOGFORMATCONSOLE 13
```

Display date and time only:

```
LOGFORMATCONSOLE 5
```

See also

LOGFORMATFILE, LOGFORMATEMS

LOGFORMATEMS

Use this parameter to control the format of the log messages that are written to EMS.

Parameter Syntax

```
LOGFORMATEMS format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

bit 1 (decimal 1)	Date
bit 2 (decimal 2)	Header (log messages a pre-fixed with "[log]")
bit 3 (decimal 4)	Time
bit 4 (decimal 8)	Milliseconds
bit 5 (decimal 16)	Process ID (name or PIN)
bit 7 (decimal 64)	Log level of message

Default

The default log format is 93 (date, time, milliseconds, process ID, and log level).

Example

Display date, time, and milliseconds only:

```
LOGFORMATEMS 13
```

Display date and time only:

```
LOGFORMATEMS 5
```

See also

LOGFORMATCONSOLE, LOGFORMATFILE

LOGFORMATFILE

Use this parameter to control the format of the log messages that are written to the log file.

Parameter Syntax

```
LOGFORMATFILE format
```

Arguments

format

A number is used to represent a bit mask that controls the format. Following are the values and their corresponding format:

bit 1 (decimal 1)	Date
bit 2 (decimal 2)	Header (log messages a pre-fixed with "[log]")
bit 3 (decimal 4)	Time
bit 4 (decimal 8)	Milliseconds
bit 5 (decimal 16)	Process ID (name or PIN)
bit 7 (decimal 64)	Log level of message

Default

The default log format is 93 (date, time, milliseconds, process ID, and log level).

Example

Display date, time, and milliseconds only:

```
LOGFORMATFILE 13
```

Display date and time only:

```
LOGFORMATFILE 5
```

See also

LOGFORMATCONSOLE, LOGFORMATEMS

LOGLEVEL

Use this parameter to control the level of detail of messages that are written to the console or log file.

Parameter Syntax

```
LOGLEVEL detail
```

Arguments

detail

A number is used to represent the level of detail desired. Following is more information about the values allowed:

- A valid number must be between 0, indicating no messages, and 100. The value of 100 indicates the maximum amount of messages. The maximum number should not to be used in production environments.
- The recommended level of detail is 30, indicating only startup and problem messages are written, or 50, specifying some usage messages are also written.

Considerations

- This parameter is retained for downward compatibility only and has been replaced by the LOGLEVELCONSOLE and LOGLEVELFILE parameters.
- If no value is set for the LOGLEVELCONSOLE or LOGLEVELFILE parameters, they will inherit their value from the LOGLEVEL parameter.
- If both LOGLEVELCONSOLE and LOGLEVELFILE parameters are assigned a value, the LOGLEVEL parameter becomes meaningless.

See also

LOGLEVELCONSOLE, LOGLEVELEMS, LOGLEVELFILE

LOGLEVELCACHE

Use this parameter to control what messages are written to the log cache.

Parameter Syntax

```
LOGLEVELCACHE detail
```

Arguments

detail

A number specifying the detail level.

Default

A default of 50 is used.

Considerations

- Using the LOGLEVELCACHE parameter allows users to set a different log level for the log messages written to the log cache than for the output written to LOGFILE.
- Writing log messages to the log cache and writing the current content to the log file sporadically as required can reduce the number of disk operations needed for logging.
- The size of the log cache can be configured.
- The content of the log cache can be written to the configured LOGFILE.
- The format of log message written to the log cache is determined by the setting of LOGFORMATFILE.

See also

LOGLEVELSIZE, LOGLEVELFILE

LOGLEVELCONSOLE

Use this parameter to control what messages are written to the log console.

Parameter Syntax

```
LOGLEVELCONSOLE detail
```

Arguments

detail

A number specifying the detail level.

Default

For downward compatibility, the default log level is taken from the parameter LOGLEVEL if present. If no LOGLEVEL parameter is present, a default of 50 is used.

Considerations

- Using the LOGLEVELCONSOLE parameter allows users to set a different log level for the output written to LOGCONSOLE than for the output written to LOGFILE.

See also

LOGCONSOLE, LOGLEVELFILE, LOGFORMATCONSOLE

LOGLEVELEMS

Use this parameter to control which messages are written to EMS.

Parameter Syntax

```
LOGLEVELEMS detail
```

Arguments

detail

A number specifying the detail level.

Default

The default value for this parameter is 20.

Considerations

- Different log levels can be used for the outputs to LOGCONSOLE, LOGEMS, and LOGFILE.

- Using the SSHCOM command interpreter, you can change parameters without having to restart SSH2.

See also

LOGEMS, LOGLEVELCONSOLE, LOGLEVELFILE, LOGFORMATEMS

LOGLEVELFILE

Use this parameter to control which messages are written to the log file.

Parameter Syntax

`LOGLEVELFILE` *`detail`*

Arguments

`detail`

A number specifying the detail level.

Default

For downward compatibility, the default log level is taken from the LOGLEVEL parameter, if present. Otherwise, a default of 50 is used.

Considerations

- Different log levels can be used for the outputs to LOGCONSOLE, LOGEMS, and LOGFILE.
- With the SSHCOM command interpreter, users can change parameters without having to restart SSH2.

See also

LOGFILE, LOGLEVELCONSOLE, LOGMAXFILELENGTH, LOGFORMATFILE

LOGMAXFILELENGTH

Use this parameter to control the maximum size of a log file.

Parameter Syntax

`LOGMAXFILELENGTH` *`length`*

Arguments

`length`

Represents the maximum log file length in kilobytes. Following are the ranges allowed:

Maximum: 40.000 or 40 MB

Minimum: 100 KB

Default

The default length is 20,000 KB.

Considerations

- After the current log file reaches the maximum size, a log rollover will occur. The current log file will be renamed by appending a number to its name. A new file with the LOGFILE name will be created for subsequent log output.

See also

- LOGFILE, LOGLEVELFILE, LOGFILERETENTION
- "Log Messages" in the "Monitoring and Auditing" chapter.

LOGMEMORY

Use this parameter to include SSH2 memory usage statistics in the log output at regular intervals.

Parameter Syntax

```
LOGMEMORY number_of_ios
```

Arguments

number_of_ios

A number that represents how many I/O operations are to be conducted before SSH2 includes its memory usage in the log output

Default

The default is 0, meaning that memory usage will not be logged.

Considerations

- Provides an easy way to correlate between memory usage of SSH2 and events in the log output. Do not use if memory usage of SSH2 is not of interest to you.

MACS

Use this parameter to specify which message authentication codes (MAC) are admissible for the SSH2 server.

Parameter Syntax

```
MACS mac [, mac, ...]
```

Arguments

mac

Specifies a MAC. Currently the following MACs are supported by SSH2:

- hmac-sha1: HMAC-SHA1 (digest length=key length=20 bytes=160 bits)
- hmac-md5: HMAC-MD5 (digest length=key length=16 bytes=128 bits)
- hmac-sha1-96: first 96 bits of HMAC-SHA1 (digest length=12 bytes=96 bits, key length=20 bytes=160 bits)
- hmac-md5-96: first 96 bits of HMAC-MD5 (digest length=12 bytes=96 bits, key length=16 bytes=128 bits)

Considerations

For details about the MACs listed above, please refer to standard SSH documentation, such as the available RFCs.

Default

If this parameter is omitted, SSH2 will accept all MACs listed above.

Example

```
MACS hmac-sha1-96
```

This will enforce the use of the hmac-sha1-96 MAC algorithm.

PARTIALSSHCOMACCESSGROUP<n>

This parameter set allows granting limited administrative SSHCOM command privileges to users that have the configured group as PRIMARY-GROUP in the Safeguard USER configuration. Admin groups with limited SSHCOM access are defined via the parameter set PARTIALSSHCOMACCESSGROUP<n> where <n> is a number between 1 and 99.

Limited administrative SSHCOM access includes viewing and altering USER records, i.e. execution of daemon mode commands INFO USER and ALTER USER. All USER attributes can be modified except the most critical ones, which are ALLOWED-AUTHENTICATIONS and SYSTEM-USER. These fields can only be modified by users with full SSHCOM access.

Additional restrictions apply depending on the setting of parameter LIFECYCLEPOLICYPUBLICUSERKEY: Users with partial SSHCOM access can specify the LIVE-DATE and EXPIRE-DATE when adding or altering a user's public key only if LIFECYCLEPOLICYPUBLICUSERKEY is set to VARIABLE.

Parameter Syntax

```
PARTIALSSHCOMACCESSGROUP<j> <group>
```

Arguments

`<group>`

A Guardian group name. All members of the group will have partial SSHCOM access.

Default

By default, none of the parameters are set, i.e. only users with full SSHCOM access can execute privileged commands.

Example

```
PARTIALSSHCOMACCESSGROUP1 admin
PARTIALSSHCOMACCESSGROUP2 super
```

Considerations

- Some of the privileged commands in SSHCOM are critical to the security of the system. Therefore granting access to other user accounts than super.super must be carefully considered.
- The parameters must be set contiguously, i.e. if one parameter PARTIALSSHCOMACCESSGROUP<p> is not defined the checking of PARTIALSSHCOMACCESSGROUP<n> parameters stops.
- This parameter set is valid whether a thawed OBJECTTYPE USER record exists in Safeguard or not. But if a user is configured with C access in the OBJECTTYPE USER record as well as included in the parameter set PARTIALSSHCOMACCESSGROUP<n>, then the user has full SSHCOM access.
- If a user is included in parameter sets PARTIALSSHCOMACCESSGROUP<n> as well as sets FULLSSHCOMACCESSUSER<i> or FULLSSHCOMACCESSGROUP<j>, then the user has full SSHCOM access.

See also

- PARTIALSSHCOMACCESSUSER<i>, FULLSSHCOMACCESSUSER<i>, FULLSSHCOMACCESSGROUP<j>, LIFECYCLEPOLICYPUBLICUSERKEY
- See table in "SSHCOM Access Summary" in section "SSHCOM Command Reference".

PARTIALSSHCOMACCESSUSER<k>

This parameter set allows granting limited administrative SSHCOM command privileges to configured users. Admin users with limited SSHCOM access are defined via the parameter set PARTIALSSHCOMACCESSUSER<k> where <k> is a number between 1 and 99.

Limited administrative SSHCOM access includes viewing and altering USER records, i.e. execution of daemon mode commands INFO USER and ALTER USER. All USER attributes can be modified but the most critical ones, which are ALLOWED-AUTHENTICATIONS and SYSTEM-USER, can only be modified by users with full SSHCOM access.

Additional restrictions apply depending on the setting of parameter LIFECYCLEPOLICYPUBLICUSERKEY: Users with partial SSHCOM access can specify the LIVE-DATE and EXPIRE-DATE when adding or altering a user's public key only if LIFECYCLEPOLICYPUBLICUSERKEY is set to VARIABLE.

Parameter Syntax

```
PARTIALSSHCOMACCESSUSER<k> <group>.<user>
```

Arguments

<group>.<user>

The Guardian logon name of the account that will have partial SSHCOM access. Logon ids and alias names are not supported.

Default

By default, none of the parameters are set, i.e. only users with full SSHCOM access can execute privileged commands.

Example

```
PARTIALSSHCOMACCESSUSER1 admin.joe
PARTIALSSHCOMACCESSUSER2 admin.jim
PARTIALSSHCOMACCESSUSER3 super.jane
```

Considerations

- Some of the privileged commands in SSHCOM are critical to the security of the system. Therefore granting access to other user accounts than super.super must be carefully considered.
- The parameters must be set contiguously, i.e. if one parameter PARTIALSSHCOMACCESSUSER<k> is not defined the checking of PARTIALSSHCOMACCESSUSER<i> parameters stops.
- This parameter set is valid whether a thawed OBJECTTYPE USER record exists in Safeguard or not. But if a user is configured with C access in the OBJECTTYPE USER record as well as mentioned in the parameter set PARTIALSSHCOMACCESSUSER<k>, then the user has full SSHCOM access.
- If a user is included in parameter sets PARTIALSSHCOMACCESSGROUP<n> as well as sets FULLSSHCOMACCESSUSER<i> or FULLSSHCOMACCESSGROUP<j>, then the user has full SSHCOM access.

See also

- PARTIALSSHCOMACCESSGROUP<n>, FULLSSHCOMACCESSUSER<i>, FULLSSHCOMACCESSGROUP<j>, LIFECYCLEPOLICYPUBLICUSERKEY
- See table in "SSHCOM Access Summary" in section "SSHCOM Command Reference".

PAUTHSUPPRESSIPADDRESS

Local authentication with password provides the remote client IP address to system procedure USER_AUTHENTICATE_ if the OS release supports this (H06.26 or later and J06.15 or later). If the IP address needs to be suppressed in USER_AUTHENTICATE_ calls, then parameter PAUTHSUPPRESSIPADDRESS must be set to TRUE.

Parameter Syntax

```
PAUTHSUPPRESSIPADDRESS TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether the IP address must be suppressed in USER_AUTHENTICATE_ calls or not. Valid values are:

- TRUE: The IP address gets suppressed.
- FALSE: The IP address is supplied.

Default

If omitted, value FALSE is the default value.

Example

```
PAUTHSUPPRESSIPADDRESS TRUE
```

PORT

Use this parameter to specify the port number a SSH2 server should listen on for incoming connections.

Parameter Syntax

`PORT` *`number`*

Arguments

number

Refers to the decimal number of a TCP/IP port.

Default

The default for this parameter is 22.

Considerations

- The ICANN manages a list of "well-known" port numbers for various protocols (see http://www.iana.org/assignments/port-numbers). 22 is the well-known port for the SSH protocol.
- The choice for the port value in your specific environment will depend on the applications already running on your NonStop systems, the ports in use, and your firewall configuration.

PROPAGATEDEFINES

This parameter controls whether SSH2 propagates defines in the SSH2 process context to newly started processes.

Parameter Syntax

`PROPAGATEDEFINES` *`TRUE|FALSE`*

Arguments

TRUE|FALSE

Specifies if SSH2 propagates defines or not. Valid values are:

- TRUE: Defines will be propagated
- FALSE: Defines will not be propagated.

Default

If omitted, PROPAGATEDEFINES will be set to TRUE. This is consistent with the behavior since introduction of define propagation.

Considerations

- The =_DEFAULTS DEFINE is always propagated to other processes regardless of the setting of the PROPAGATEDEFINES parameter.

Example

```
PROPAGATEDEFINES FALSE
```

See also

PTCPIPFILTERKEY

PTCPIPFILTERKEY

Use this parameter to specify a filter key to enable round-robin filtering with parallel library TCP/IP or TCP/IPV6.

Parameter Syntax

```
PTCPIPFILTERKEY password | *
```

Arguments

password

A password that serves as a key to enable round-robin filtering of multiple instances of SSH2 servers listening on the same port. The password will override the value of the DEFINE =PTCPIP^FILTER^KEY, which may have been passed to SSH2 at startup.

*

No filter key will be set. However, any DEFINE =PTCPIP^FILTER^KEY passed to SSH2 at startup will remain in effect.

Default

The default for this parameter is *.

Considerations

- Use this parameter to enable round-robin filtering for multiple SSH2 servers configured to run as generic processes. This can also be achieved by adding the define =PTCPIP^FILTER^KEY for the generic process (possible since G06.28/H06.06).
- In case the define =PTCPIP^FILTER^KEY causes unwanted behaviour, it is possible to disable the propagation of defines completely, see parameter PROPAGATEDEFINES

See also

PROPAGATEDEFINES

PTCPIPFILTERTCPPORTS

Use this parameter to limit port sharing in case round-robin filtering is enabled.

Parameter Syntax

```
PTCPIPFILTERTCPPORTS Pstartport.Pendport | *
```

Arguments

Pstartport.Pendport

A port range from startport to endport that restricts shared ports to the configured port range. The configuration is only effective if round-robin is enabled, i.e. if either the DEFINE =PTCPIP^FILTER^KEY or the SSH2 parameter PTCPIPFILTERKEY is set.

*

Shared ports will not be limited. However, any DEFINE =PTCPIP^FILTER^TCP^PORTS passed to SSH2 at startup will remain in effect.

Default

The default for this parameter is *.

Considerations

- Use this parameter to limit shared ports when round-robin filtering is enabled for multiple SSH2 servers configured as generic processes. This can also be achieved by adding the define =PTCPIP^FILTER^TCP^PORTS for the generic process (possible since G06.28/H06.06).
- In case the define =PTCPIP^FILTER^TCP^PORTS causes unwanted behaviour, it is possible to disable the propagation of defines completely, see parameter PROPAGATEDEFINES

See also

PROPAGATEDEFINES

PTYSERVER

Use this parameter to specify the name of an STN process serving as a pseudo terminal (PTY) server.

Parameter Syntax

```
PTYSERVER processname
```

Arguments

processname

Specifies the name of an STN process.

Default

The default for this parameter is $PTY.

Considerations

- Value is used as default value for USER attribute PTY-SERVER.
- Please refer to the "Enabling Full TTY Access" section for details.

RECORDDELIMITER

Use this SFTP related parameter to define the end-of-record indicator in files transferred from a remote host to a structured file on NonStop. The parameter is relevant if the SFTP server on NonStop is used for file transfer or if the SFTP client on NonStop is used and the SFTP command ASCII is not issued before the file transfer (i.e. the transfer is made in binary mode).

Parameter Syntax

```
RECORDDELIMITER LF|CR|CRLF|ANY
```

Arguments

LF

End of Record is indicated by an LF (hexadecimal 0A, escape character \n)

CR

End of Record is indicated by a CR (hexadecimal 0D, escape character \r).

CRLF

End of Record is indicated by a CR followed by an LF (hexadecimal 0D0A, escape characters \n\r).

ANY

End of Record can be CR (0D), LF (0A) or CRLF (0D0A).

Considerations:

- In SSH2 versions before 0085 the default processing was ANY. If files transferred and directly stored in a structured NonStop use other end-of-record delimiters, i.e. CR (0D) or CRLF (0D 0A), then the parameter RECORDDELIMITER must now be set with a value of ANY.
- The SFTP client on NonStop supports the command ASCII with additional options (see chapter "SFTP Client Command Reference") allowing setting the accepted end of record delimiter (ASCII MAC corresponds to CR, ASCII DOS to CRLF and ASCII UNIX to LF). That is, for the SFTP client the setting of parameter RECORDDELIMITER is just the default setting, which can be overwritten using the SFTP client command ASCII.
- The characters LF and CR cannot occur inside the record data if the value of RECORDDELIMITER is ANY. The character LF (0A) is not allowed in the record data if the parameter is set to LF. The character CR (0D) is not allowed in the record data if the parameter is set to CR.
- The record delimiter is a local setting, i.e. there is no negotiation of the record delimiter between ssh client and ssh server in the supported sftp protocol. The entity reading from a structured file or Guardian edit file must add the record delimiter to each record read. The entity writing to a structured file or Guardian edit file must split the received data accordingly and remove the record delimiter before writing the record.

Default

The default for this parameter is LF.

RESTRICTIONCHECKFAILEDDEFAULT

Use this parameter to define the outcome of restriction checks (related to RESTRICTION-PROFILE) in cases in which no USER record was found for the Guardian user starting an outgoing SSH connection.

Parameter Syntax

```
RESTRICTIONCHECKFAILEDDEFAULT TRUE|FALSE
```

Arguments

TRUE

Restriction checks will fail if a USER record could not be found.

FALSE

Restriction checks will not fail if a USER record could not be found.

Default

The default for this parameter is FALSE.

SAFEGUARD-PASSWORD-REQUIRED

For G-Series and H-Series RVU prior to H06.11, set this parameter according to the Safeguard PASSWORD-REQUIRED configuration.

Parameter Syntax

```
SAFEGUARD-PASSWORD-REQUIRED TRUE|FALSE
```

Arguments

TRUE

Safeguard PASSWORD-REQUIRED is ON.

FALSE

Safeguard PASSWORD-REQUIRED is OFF.

Considerations

- G-Series and H-Series RVU prior to H06.11 do not support PRIV logon of a Safeguard ALIAS. Hence, SSH2 can only impersonate an ALIAS if a password is provided. If this parameter is set to TRUE, SSH2 will always request that users mapped to an ALIAS perform password authentication, even after a successful public key authentication.
- Do not set this parameter for H06.11 RVU or later.

Default

If omitted, the default will be FALSE.

Example

```
SAFEGUARD-PASSWORD-REQUIRED TRUE
```

SFTPALLOWGUARDIANCD

Use this parameter to enable the usage of a Guardian style CD command with SFTPSERV.

Parameter Syntax

```
SFTPALLOWGUARDIANCD TRUE|FALSE
```

Arguments

TRUE

SFTP clients can use Guardian-style CD commands, such as "CD $data05.mysvol".

FALSE

SFTP clients can only use Unix-style CD commands.

Considerations

- The mechanism for resolving Guardian-style sub-volume names may cause problems with some SFTP clients, such as FileZilla.
- The CD command with Guardian volume and sub-volume only works in the Guardian name space (path starts with /G). Switching from OSS name space to Guardian name space requires either to put /G in front of the sub-volume (e.g. cd /G/$us.temp) or to issue a separate cd /G command. This is required only once. When in Guardian name space a simple cd <sub-volume>, e.g. cd $us.temp, is sufficient.

Default

If omitted, the default will be FALSE.

Example

```
SFTPALLOWGUARDIANCD TRUE
```

SFTPCPUSET

This parameter allows configuring the default set of CPUs the SSH2 process starts SFTPSERV user processes in.

Parameter Syntax

```
SFTPCPUSET cpu-set
```

Arguments

cpu-set

A comma separated list of CPU numbers or CPU number ranges defining allowed CPUs.

Default

If omitted, SSH2 will start all SFTPSERV processes in the CPU the SSH2 process is running in unless the USER record specifies a different CPU set for a specific user via attribute SFTP-CPU-SET.

Example

```
SFTPCPUSET 2-4,7,10,13-15
```

Considerations

- A value configured in USER attribute SFTP-CPU-SET has higher priority than the value defined in the SSH2 parameter SFTPCPUSET.

See also

CPUSET

SFTPDISPLAYGUARDIAN

Use this parameter to control file name format (Guardian or OSS) in SFTP informational messages like "Uploading ..." and "Fetching ...". Alternately, define =SFTP^DISPLAY^GUARDIAN can be set; define overrides PARAM.

Parameter Syntax

```
SFTPDISPLAYGUARDIAN TRUE|FALSE
```

Arguments

TRUE

Guardian file name format is used.

FALSE

File names are displayed in standard ssh format (Unix style with OSS prefix /G or /E).

Default

The default value is FALSE.

Considerations

- Note that the default Unix style was introduced in SPR T0801^AAS to better conform to the SFTP standard; before that, the Guardian style was the default.

SFTPEDITLINEMODE

Use this parameter to control the handling of Guardian edit lines that are too long when a file transfer is made to a Guardian edit file on the NonStop server.

Parameter Syntax

```
SFTPEDITLINEMODE none | cut | wrap
```

Arguments

none

No special handling is done. A long line is treated as an error.

cut

The long line will be cut to ensure a maximum line length of 239 characters.

wrap

The long line will be wrapped, i.e. the first part of the line will be written in 239 character chunks until less than 240 characters are left, which will be written last.

Default

The default value is none.

Considerations

- The setting of this parameter is only relevant if parameter SFTPEDITLINESTARTDECIMALINCR is set to a number between 0 and 99999999.
- This parameter is only considered when a Guardian edit file is written, i.e. either if a remote sftp client issues a put command to the SSH2 server on NonStop specifying a Guardian destination file with code 101 or if a sftp client on a NonStop server issues a get command specifying a local Guardian destination file with file code 101.
- If a get command is executed by a sftp client on the NonStop server, then the parameter must be set in the environment of the sftp client (as PARAM for SFTP running in the Guardian environment or as environment variable for SFTPOSS running in the OSS environment).
- The parameter SFTPEDITLINEMODE defines the default behavior when Guardian edit files are created. The handling of lines that are too long can be altered by issuing the command ASLINEMODE at the NonStop SFTP client prompt. The ASLINEMODE command takes one of the values *none*, *cut* and *wrap* as parameter.

See also

SFTPEDITLINENUMBERDECIMALINCR, SFTPEDITLINESTARTDECIMALINCR

SFTPEDITLINENUMBERDECIMALINCR

Use this parameter to define the decimal increment used to calculate the next Guardian edit line number when a file transfer is made to a Guardian edit file on the NonStop™ server.

Parameter Syntax

```
SFTPEDITLINENUMBERDECIMALINCR <number>
```

Arguments

<number>

The value is 1000 times the increment. See documentation for Guardian procedure call INCREMENTEDIT.

Default

The default value is 1000, i.e. the line numbers are incremented by 1)

Examples

Increment by 0.003:

```
SFTPEDITLINENUMBERDECIMALINCR 3
```

Increment by 0.1:

```
SFTPEDITLINENUMBERDECIMALINCR 100
```

Considerations

- The setting of this parameter is only relevant if parameter SFTPEDITLINESTARTDECIMALINCR is set to a number between 0 and 99999999.
- Previously, all Guardian edit files were written starting with line number 1 and increment 1.000, which allowed a maximum of 99999 lines. This behavior is still the default.
- The default increment (1.000) is used for all lines less than the value of parameter SFTPEDITLINESTARTDECIMALINCR. In order to get the same result as the NonStop FTP server the parameter SFTPEDITLINENUMBERDECIMALINCR must be set to 100 and the value of SFTPEDITLINESTARTDECIMALINCR to 40000000.
- This parameter is only considered when a Guardian edit file is written, i.e. either if a remote sftp client issues a put command to the SSH2 server on NonStop specifying a Guardian destination file with code 101 or if a sftp client on a NonStop server issues a get command specifying a local Guardian destination file with file code 101.
- If a get command is executed by a sftp client on the NonStop server, then the parameter must be set in the environment of the sftp client (as PARAM for SFTP running in the Guardian environment or as environment variable for SFTPOSS running in the OSS environment).

See also

SFTPEDITLINEMODE, SFTPEDITLINESTARTDECIMALINCR

SFTPEDITLINESTARTDECIMALINCR

This parameter controls at which line number the decimal increment defined by parameter SFTPEDITLINENUMBERDECIMALINCR starts.

Parameter Syntax

```
SFTPEDITLINESTARTDECIMALINCR <number>
```

Arguments

<number>

The value is 1000 times the line number.

Default

The default value is -1, i.e. decimal increment is not used.

Examples

Start decimal increment at line number 40000:

```
SFTPEDITLINENUMBERDECIMALINCR 40000000
```

Start decimal increment at line number 0.000:

```
SFTPEDITLINENUMBERDECIMALINCR 0
```

Considerations

- The setting of this parameter is only relevant if parameter SFTPEDITLINESTARTDECIMALINCR is set to a number between 0 and 99999999.
- Previously, all Guardian edit files were written starting with line number 1 and increment 1.000, which allowed a maximum of 99999 lines. This behavior is still the default.
- The default increment (1.000) is used for all lines less than the value of parameter SFTPEDITLINESTARTDECIMALINCR. In order to get the same result as the NonStop FTP server the parameter SFTPEDITLINENUMBERDECIMALINCR must be set to 100 and the value of SFTPEDITLINESTARTDECIMALINCR to 40000000.
- Setting SFTPEDITLINESTARTDECIMALINCR 0 and SFTPEDITLINENUMBERDECIMALINCR to 1 allows for the maximum possible number of lines in Guardian edit files.
- This parameter is only considered when a Guardian edit file is written, i.e. either if a remote sftp client issues a put command to the SSH2 server on NonStop™ specifying a Guardian destination file with code 101 or if a sftp client on a NonStop server issues a get command specifying a local Guardian destination file with file code 101.
- If a get command is executed by a sftp client on the NonStop server, then the parameter must be set in the environment of the sftp client (as PARAM for SFTP running in the Guardian environment or as environment variable for SFTPOSS running in the OSS environment).

See also

SFTPEDITLINEMODE, SFTPEDITLINENUMBERDECIMALINCR

SFTPENHANCEDERRORREPORTING

Use this parameter to control the amount of information displayed if an error occurs in an SFTP session.

Parameter Syntax

```
SFTPENHANCEDERRORREPORTING <detail>
```

Arguments

<detail>

The level of details. Possible values: 0, 1 and 2. For value 0 the same level of detail gets produced as before introduction of parameter SFTPENHANCEDERRORREPORTING. Value 1 means increased detail level, and 2 is the maximum detail level.

Considerations

- The parameter can be set set for the SSH2 process (checked by the SFTP server) and for SFTP clients.
- For SFTP clients, either PARAM (SFTP) or environment variable (SFTPOSS) must be used to configure the parameter.
- There are errors where additional details are not (yet) available.

Default

If omitted, value 0 is the default value.

Example

```
SFTPENHANCEDERRORREPORTING 1
```

SFTPEXCLUSIONMODEREAD

Use this parameter to set the exclusion mode of structured files that are opened for read via system procedure FILE_OPEN_().

Parameter Syntax

```
SFTPEXCLUSIONMODEREAD <exclusion>
```

Arguments

<exclusion>

The file open exclusion mode for read operations. Valid values are SHARED, EXCLUSIVE and PROTECTED

Considerations

- If a file is open for write by anther process (shared or protected) and this file is to be read by SFTP or SFTPSERV, then reading this file will only fail if parameter is set to a different value than SHARED. It can be required to force a failure in this scenario to ensure the process writing the file closes the file before the file transfer.
- If a get command is executed by a sftp client on the NonStop server, then the parameter must be set in the environment of the sftp client (as PARAM for SFTP running in the Guardian environment or as environment variable for SFTPOSS running in the OSS environment).

Default

If omitted, value SHARED will be used, which was the value used prior to adding parameter SFTPEXCLUSIONMODEREAD.

Example

```
SFTPEXCLUSIONMODEREAD EXCLUSIVE
```

SFTPIDLETIMEOUT

Use this parameter to control how long SFTPSERV keeps running without any SFTP protocol traffic before terminating itself.

Parameter Syntax

```
SFTPIDLETIMEOUT <seconds>
```

Arguments

<seconds>

The time in seconds the SFTPSERV waits after the last SFTP command before it stops serving the client.

Considerations

- The SFTP client will not be able to issue further SFTP commands.

Default

If omitted, there is no SFTP idle timeout. The SFTPSERV will be running until the STP client ends the session.

Example

```
SFTPIDLETIMEOUT 180
```

SFTPMAXEXTENTS

Use this parameter to specify the MAXEXTENTS value for files that are created on the NonStop system.

Parameter Syntax

```
SFTPMAXEXTENTS maxextents
```

Arguments

maxextents

Specifies the value to be used.

Considerations

- The value can be overridden in "put" and "get" commands using the extended syntax described in "SFTP Client Reference" chapter, in the section entitled "Extended Syntax for Creation of New Guardian Files".

Default

If omitted, SSH2 will use a value of 900.

Example

```
SFTPMAXEXTENTS 950
```

SFTPPRIMARYEXTENTSIZE

Use this parameter to specify the primary extent size for files that are created on the NonStop system.

Parameter Syntax

```
SFTPPRIMARYEXTENTSIZE extsize
```

Arguments

extsize

Specifies the value to be used.

Considerations

- The value can be overridden in "put" and "get" commands using the extended syntax described in the "SFTP client reference" chapter, in the section entitled "Extended Syntax for Creation of New Guardian Files".

Default

If omitted, SSH2 will use a value of 2.

Example

```
SFTPPRIMARYEXTENTSIZE 10
```

SFTPREALPATHFILEATTRIBUTEECHOED

Enables or disables the echoing of file attributes added to file names. Some remote SFTP clients call realpath() against the SFTP server for every remote file mentioned in a get or put command. By default, any file attributes added to a file get stripped by this call. The remote SFTP clients in question then use the value returned by realpath() for the actual remote file access, i.e. without the file attributes a remote user had specified.

Parameter Syntax

```
SFTPREALPATHFILEATTRIBUTEECHOED TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether the file attributes attached to a file name get echoed by the SFTP server, i.e. returned to the SFTP client:

- TRUE: File attributes will be echoed by realpath function.

- FALSE: File attributes will be stripped by realpath function.

Default

If omitted, SSH2 will use value FALSE.

Example

```
SFTPREALPATHFILEATTRIBUTEECHOED TRUE
```

Considerations

- One SFTP client that is known to call realpath() before accessing the remote file is PuTTY. Special processing has been implemented for PuTTY: The SFTP server checks the client version string to detect a PuTTY client. If a PuTTY client was detected the file attributes will be echoed independently of the setting of parameter SFTPREALPATHFILEATTRIBUTEECHOED
- Parameter SFTPREALPATHFILEATTRIBUTEECHOED needs to be set to TRUE only for other SFTP clients that call realpath() before accessing the remote file via put or get command.

SFTPSECONDARYEXTENTSIZE

Use this parameter to specify the secondary extent size for files that are created on the NonStop system.

Parameter Syntax

```
SFTPSECONDARYEXTENTSIZE extsize
```

Arguments

extsize

Specifies the value to be used.

Considerations

- The value can be overridden in "put" and "get" commands using the extended syntax described in "Extended Syntax for Creation of New Guardian Files" section of the "SFTP Client Reference" chapter.

Default

If omitted, SSH2 will use a value of 100.

Example

```
SFTPSECONDARYEXTENTSIZE 200
```

SFTPUPSHIFTGUARDIANFILENAMES

Use this parameter to enforce uppercase characters for Guardian file names sent using the "mput" command from a NonStop server to a remote ssh server.

Parameter Syntax

```
SFTPUPSHIFTGUARDIANFILENAMES TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether the remote target file names are upshifted when Guardian files are transferred using the "mput" command:

- TRUE: Target file names will be upshifted.

- FALSE: Target file names will be downshifted.

Default

If omitted, SSH2 will use a value FALSE. The resulting behavior is the same as before this parameter was added.

Example

```
SFTPUPSHIFTGUARDIANFILENAMES TRUE
```

Considerations

- If the parameter is used as SSH2 parameter with value TRUE, then all Guardian file names displayed by the ls command appear in upper case. The SSH2 parameter is relevant for incoming connections.
- For outgoing connections the parameter must be set as PARAM for SFTP and as environment variable for SFTPOSS.
- If the value is set to TRUE the file template in the "mput" command specifying the local files to be transferred must consist of upper case characters. Otherwise an error "file not found" will be returned.

SHELLENVIRONMENT

Set default value for USER attribute SHELL-ENVIRONMENT, used when the USER attribute is not configured. The configured script is executed for non-login shells and is important to prepare the shell environment (e.g. PATH variable) for non-login shells, which use a different shell initialization than login shells.

Parameter Syntax

SHELLENVIRONMENT *shell-script*

Arguments

shell-script

a shell script with full path information that will be executed for non-login shells to preare the shell environment.

Considerations

- The configured value is only used if the USER record does not have a value configured for attribute SHELL-ENVIRONMENT

Default

If omitted, SHELLENVIRONMENT is empty.

Example

```
AUTOADDAUTHPRINCIPAL /etc/nonloginProfile
```

See also

Section "To Connect a Remote SCP Client to the NonStop Server".

SOCKETKEEPALIVE

Use this parameter to specify whether keep-alive messages should be sent to the TCP/IP sockets of established links.

Parameter Syntax

SOCKETKEEPALIVE *mode*

Arguments

mode

- 1 (on) for sending keep alive messages
- 0 (off) no messages are sent

Default

By default, keep alive messages are sent (1).

SOCKETRCVBUF

Use this parameter to control the size of the TCP/IP receive buffer. When setting this parameter to a non-zero value the specified parameter is used on a socket level.

Parameter Syntax

`SOCKETRCVBUF` *`bytes`*

Arguments

bytes

A number representing the size of the TCP/IP receive buffer in bytes. A value of 0 means the receive buffer size configured in the TCP/IP process is used.

Considerations

- Setting this parameter to a higher value can increase throughput when transferring files. Normally the value configured in the TCP/IP process is sufficiently high.

Default

The default is 0.

SOCKETSNDBUF

Use this parameter to control the size of the TCP/IP send buffer. When setting this parameter to a non-zero value the specified parameter is used on a socket level.

Parameter Syntax

`SOCKETRCVBUF` *`bytes`*

Arguments

bytes

A number representing the size of the TCP/IP send buffer in bytes. A value of 0 means the send buffer size configured in the TCP/IP process is used.

Considerations

- Setting this parameter to a higher value can increase throughput when transferring files. Normally the value configured in the TCP/IP process is sufficiently high.

Default

The default is 0.

SOCKTCPMINRXMT

Use this parameter to control the minimum time for TCP retransmission timeout. When setting this parameter to a non-zero value the specified parameter is used on socket level.

Parameter Syntax

```
SOCKTCPMINRXMT time
```

Arguments

time

A number representing the minimum time for TCP retransmission timeout. A value of 0 means the minimum time for TCP retransmission timeout configured in the TCP/IP monitor process is used.

Considerations

- Normally the value configured on TCP/IP monitor process level (TCP-MIN-REXMIT-TIMEOUT) should be sufficient, i.e. the default value should be used for parameter SOCKTCPMINRXMT. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.
- The Cluster I/O Protocols (CIP) subsystem does not support the corresponding socket option TCP_MINRXMT, i.e. the default value must be used for parameter SOCKTCPMINRXMT if CIP is involved. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.

Default

The default is 0.

SOCKTCPMAXRXMT

Use this parameter to control the maximum time for TCP retransmission timeout. When setting this parameter to a non-zero value the specified parameter is used on socket level.

Parameter Syntax

```
SOCKTCPMAXRXMT time
```

Arguments

time

A number representing the maximum time for TCP retransmission timeout. A value of 0 means the maximum time for TCP retransmission timeout configured in the TCP/IP monitor process is used.

Considerations

- Normally the value configured on TCP/IP monitor process level (TCP-MAX-REXMIT-TIMEOUT) should be sufficient, i.e. the default value should be used for parameter SOCKTCPMAXRXMT. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.
- The Cluster I/O Protocols (CIP) subsystem does not support the corresponding socket option TCP_MAXRXMT, i.e. the default value must be used for parameter SOCKTCPMAXRXMT if CIP is involved. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.

Default

The default is 0.

SOCKTCPRXMTCNT

Use this parameter to control the maximum number of continuous retransmissions prior to dropping a TCP connection. When setting this parameter to a non-zero value the specified parameter is used on socket level.

Parameter Syntax

```
SOCKTCPRXMTCNT count
```

Arguments

count

A number representing the maximum number of continuous retransmissions prior to dropping a TCP connection. A value of 0 means the maximum number of continuous retransmissions prior to dropping a TCP connection configured in the TCP/IP monitor process is used.

Considerations

- Normally the value configured on TCP/IP monitor process level should be sufficient, i.e. the default value should be used for parameter SOCKTCPRXMTCNT. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.
- The Cluster I/O Protocols (CIP) subsystem does not support the corresponding socket option TCP_RXMTCNT, i.e. the default value must be used for parameter SOCKTCPRXMTCNT if CIP is involved. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.

Default

The default is 0.

SOCKTCPTOTRXMTVAL

Use this parameter to control the maximum continuous time spent retransmitting without receiving an acknowledgement from the other endpoint. When setting this parameter to a non-zero value the specified parameter is used on socket level.

Parameter Syntax

```
SOCKTCPTOTRXMTVAL time
```

Arguments

time

A number representing the maximum time for TCP retransmission timeout. A value of 0 means the maximum continuous time spent retransmitting without receiving an acknowledgement from the other endpoint configured in the TCP/IP monitor process is used.

Considerations

- Normally the value configured on TCP/IP monitor process level should be sufficient, i.e. the default value should be used for parameter SOCKTCPTOTRXMTVAL. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.
- The Cluster I/O Protocols (CIP) subsystem does not support the corresponding socket option TCP_TOTRXMTVAL, i.e. the default value must be used for parameter SOCKTCPTOTRXMTVAL if CIP is involved. See document "HP NonStop TCP/IPv6 Configuration and Management Manual" for details.

Default

The default is 0.

SSHAUTOKEXBYTES

Use this parameter to control the frequency of automatic key re-exchange in SSH sessions.

Parameter Syntax

```
SSHAUTOKEXBYTES bytes
```

Arguments

bytes

Provides a number representing the amount of bytes after which a key re-exchange should be initiated. A value of 0 disables key re-exchange based on data volume.

Default

The default is 1073741824 (1GB). This is the value recommended in RFC 4253.

See also

SSHAUTOKEXTIME

SSHAUTOKEXTIME

Use this parameter to control the frequency of automatic key re-exchange in SSH sessions.

Parameter Syntax

```
SSHAUTOKEXTIME seconds
```

Arguments

seconds

Specifies the intervals between key re-exchanges in seconds. A value of 0 disables key re-exchange based on time intervals.

Default

The default is 3600 (1 hour). This is the value recommended in RFC 4253.

See also

SSHAUTOKEXBYTES

SSHCTL

Use this parameter to specify the filename of the user database file.

Parameter Syntax

```
SSHCTL filename
```

Arguments

filename

Specifies the name of the user database file.

Considerations

- The user data base stores information about remote users accessing the NonStop system. The user database is stored in a single ENSCRIBE file and maintained through the SSHCOM command interpreter. For more details of the user database, please see the "The SSH User Database" chapter.

- In order to prevent unauthorized access, the user database is stored in a proprietary format and encrypted. The database file is secured as "----".
- The customer name configured via parameter CUSTOMER or, if that does not exist, the customer name held within the license file for the SSH2 program is used as an input for host-based key encryption. When you plan to duplicate the host key and user database onto other NonStop systems (such as a disaster recovery system), you need to make sure the parameter CUSTOMER or the license file of that other system has the same customer name in it. Otherwise, the host key file and user data base cannot be used on the other system. If you purge the HOSTKEY and SSHCTL files and restart the SSH2 process, a new HOSTKEY and SSHCTL file will be created using either the value of parameter CUSTOMER or, if that does not exist, the customer name from the license file.

  Although a license file is no longer required for NonStop SSH on H and J operating systems, any existing HOSTKEY and SSHCTL file requires the customer name that was used to create the file. If a license file exists, the customer name will be extracted from that file (entry SSH2.customer), unless parameter CUSTOMER is set in which case the value of CUSTOMER is used. If a license file does not exist and an existing HOSTKEY or SSHCTL file is accessed, the parameter CUSTOMER must be set to the original value for the customer name.
- Multiple instances of the SSH2 object can share the same user database or use different user databases.
- If the SSHCTL parameter points to a non-existing file, a new and empty user database will be created on startup. SSH2 will abort at startup if the SSH database does not exist, parameter SSHCTLAUDIT is true but the SSHCTL parameter value (or its default value) does not reference an audited disk. An appropriate error message is issued in this case. The parameters SSHCTLAUDIT and SSHCTL must be set consistently to avoid this abend: If SSHCTLAUDIT is true at time of ssh database creation, then SSHCTL must point to a volume that is audited.
- The user database can be created as an audited file, allowing automatic replication of changes to another system, as well as roll-back of changes through TMF. See the "SSHCTLAUDIT" section for details.
- If multiple SSH2 processes started from the same subvolume but used for different purposes, then not only separate SSH database files (configured via SSHCTL) but separate host key files (configured via HOSTKEY) should be configured. Example: SSH for maintenance and public network.

Default

If omitted, SSH2 will use a file name of SSHCTL.

Example

```
SSHCTL $SYSTEM.SSH2.USERDB1
```

See also

- CUSTOMER

SSHCTLAUDIT

Use this parameter to specify whether a newly created user database will be set up as an audited file.

Parameter Syntax

SSHCTLAUDIT *TRUE|FALSE*

Arguments

TRUE|FALSE

Specifies whether a new user data base file will be set up as an audited file. Following are the possible arguments:

- o TRUE: file will be created as audited file.

- FALSE: file will not be created as audited file.

Considerations

- See parameter "SSHCTL" for details about the user data base.

Default

If omitted, SSH2 will use a value of TRUE.

Example

```
SSHCTLAUDIT FALSE
```

SSHKEEPALIVETIME

Use this parameter to control the frequency of SSH "keepalive" messages.

Parameter Syntax

`SSHKEEPALIVETIME` *`seconds`*

Arguments

`seconds`

Defines the idle time in seconds after which an SSH_MSG_IGNORE message is sent to the remote client. A value of 0 disables sending SSH_MSG_IGNORE messages.

Default

The default is 60 (1 minute).

Considerations

- SSHKEEPALIVETIME controls "keepalive" messages on the secure shell protocol level, while SOCKETKEEPALIVE controls whether keepalive messages should be enabled on TCP socket level.
- Sending these messages on idle sessions is an additional measure of protection against advanced traffic analysis techniques.

STOREDPASSWORDSONLY

Use this SSH2 parameter to disable the prompt for password during user authentication with method password in outgoing connections, assuming that the password is stored in the database.

Parameter Syntax

`STOREDPASSWORDSONLY` *`TRUE|FALSE`*

Arguments

`TRUE|FALSE`

Specifies whether password prompt is suppressed or not. Following are the possible arguments:

- TRUE: Password prompt is suppressed. If the password cannot be found in the SSHCTL database, then the password authentication will fail.
- FALSE: Users will be prompted for the password if that was not found in the SSHCTL database.

Default

The default is FALSE. The default behavior is therefore the same as before this parameter was introduced.

Considerations

- This parameter is only relevant for outgoing connections, i.e. with ssh clients SSH[OSSS] and SFTP[OSS] running on a NonStop™ server.
- In a scenario of ssh clients running in batch mode where password authentication is a requirement the password prompt does not make sense.

STRICTHOSTKEYCHECKING

This option controls whether to restrict client access to remote systems to only those cases in which the host's public key is explicitly configured as a KNOWNHOST entity in the SSHCTL.

Parameter Syntax

```
STRICTHOSTKEYCHECKING TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether host key of remote hosts must be preconfigured in SSHCTL. Following are the possible arguments:

- o TRUE: Access to unknown hosts will be denied.
- o FALSE: Users will be prompted if they want to continue a connection to an unknown host and add the host's public key as a KNOWNHOST entity to the SSHCTL.

Considerations

- KNOWNHOST entities can be configured using SSHCOM.

Default

If this option is omitted, SSH2 will use a value of TRUE.

Example

```
STRICTHOSTKEYCHECKING FALSE
```

SUBNET

Use this parameter to specify the TCP/IP process(es) an SSH2 process should listen on for incoming connections.

Parameter Syntax

```
SUBNET tcpip-process-name [,tcpip-process-name, ...]
```

Arguments

tcpip-process-name

Name of an existing TCP/IP process in your system.

Default

If omitted, the SSH2 process will be bound to "$ZTC0".

Example

```
SUBNET $ZTC03
```

Considerations

- If you added a DEFINE =TCPIP^PROCESS^NAME to the TACL environment you use to start SSH2, this setting will override the SUBNET parameter.
- If you use parallel library TCPIP and want to share identical ports across multiple instances of SSH2, you need to add an identical DEFINE to all instances sharing that port as in the following example:

```
ADD DEFINE =PTCPIP^FILTER^KEY, class map, file A1234
```

- If parameter is set via PARAM and a comma separated list is defined, then the list must be enclosed in double quotes.

See also

INTERFACE, INTERFACEOUT

SUPPRESSCOMMENTINSSHVERSION

Use this parameter to suppress the 'comments' field in SSH protocol version exchanged between ssh server and ssh client. The format of the ssh protocol version is defined in RFC 4253. The 'comments' field is defined as optional.

Parameter Syntax

```
SUPPRESSCOMMENTINSSHVERSION TRUE|FALSE
```

Arguments

TRUE|FALSE

Specifies whether comment part in the ssh protocol version is suppressed or not:

- TRUE: Comment part will be suppressed.
- FALSE: Comment part will not be suppressed.

Default

If omitted, the SSH2 process will include the comment part as done in the previous release, i.e. default value is FALSE.

Considerations

- RFC 4253 defines that client and server ssh protocol version string must be exchanged in clear text. This could give away information about implementation details, which might be seen as a vulnerability. Using this parameter only the optional part of the protocol version string can be suppressed.
- On the other hand, the comments part may indicate specific capabilities of an implementation, i.e. can be helpful information for the remote system.

TCPIPHOSTFILE

Use this parameter as an alternative to setting a DEFINE =TCPIP^HOST^FILE.

Parameter Syntax

```
TCPIPHOSTFILE filename
```

Arguments

filename

Specifies the name of the TCPIP host file to be used by SSH2. The file name will override the value of the DEFINE =TCPIP^HOST^FILE parameter, which may have been passed to SSH2 at startup.

*

Indicates no host file will be set. However, any DEFINE =TCPIP^HOST^FILE passed to SSH2 at startup will remain in effect.

Default

The default for this parameter is *.

Considerations

- Use this parameter to pass the value for the DEFINE =TCPIP^HOST^FILE to SSH2 servers configured as generic processes. This can also be achieved by adding the define =TCPIP^HOST^FILE for the generic process (possible since G06.28/H06.06).
- In case the define =TCPIP^HOST^FILE causes unwanted behaviour, it is possible to disable the propagation of defines completely, see parameter PROPAGATEDEFINES
- An entry TCPIPHOSTFILE $system.ztcpip.empty has been added to the SSH2 configuration file for the maintenance LAN (file SSHmCFG starting with H06.25/J06.14 to bypass DNS lookup. This solves a problem of a 40 seconds delay when executing an SSH command against a CLIM (e.g. using CLIMCMD) due to unresolved DNS lookups. Although this is a problem with the DNS configuration, the above workaround has been put into place to prevent these delays. Name resolution delays are now detected during SSH2 startup and a warning message will be issued.

See also

PROPAGATEDEFINES

TCPIPNODEFILE

Use this parameter as an alternative to setting a DEFINE =TCPIP^NODE^FILE.

Parameter Syntax

```
TCPIPNODEFILE filename
```

Arguments

filename

Specifies the name of the TCPIP node file to be used by SSH2. The filename will override the value of the DEFINE =TCPIP^NODE^FILE, which may have been passed to SSH2 at startup.

*

Means no node file will be set. However, any DEFINE =TCPIP^NODE^FILE passed to SSH2 at startup will remain in effect.

Default

The default for this parameter is *.

Considerations

- Use this parameter to pass the value for the DEFINE =TCPIP^NODE^FILE to SSH2 servers configured as generic processes. This can also be achieved by adding the define =TCPIP^NODE^FILE for the generic process (possible since G06.28/H06.06).
- In case the define =TCPIP^NODE^FILE causes unwanted behaviour, it is possible to disable the propagation of defines completely, see parameter PROPAGATEDEFINES.

See also

PROPAGATEDEFINES

TCPIPRESOLVERNAME

Use this parameter as an alternative to setting a DEFINE =TCPIP^RESOLVER^NAME.

Parameter Syntax

```
TCPIPRESOLVERNAME filename
```

Arguments

filename

Specifies the name of the RESCONF file to be used by SSH2. The filename will override the value of the DEFINE =TCPIP^RESOLVER^NAME, which may have been passed to SSH2 at startup.

*

Indicates no RESCONF file will be set. However, any DEFINE =TCPIP^RESOLVER^NAME passed to SSH2 at startup will remain in effect.

Default

The default for this parameter is *.

Considerations

- Use this parameter to pass the value for the DEFINE =TCPIP^RESOLVER^NAME parameter to SSH2 servers configured as generic processes. This can also be achieved by adding the define =TCPIP^RESOLVER^NAME for the generic process (possible since G06.28/H06.06).
- In case the define =TCPIP^RESOLVER^NAME causes unwanted behaviour, it is possible to disable the propagation of defines completely, see parameter PROPAGATEDEFINES.

See also

PROPAGATEDEFINES

USETEMPLATESYSTEMUSER

The SYSTEM-USER of the template user is used for an automatically added user if the Boolean parameter USETEMPLATESYSTEMUSER is TRUE. The value of USETEMPLATESYSTEMUSER is only relevant in case AUTOADDSYSTEMUSERS is set to TRUE and AUTOADDSYSTEMUSERSLIKE is configured (defining the template USER record). This allows the addition of users with the same (dummy) Guardian user ID or with the SYSTEM-USER value of *NONE*.

Parameter Syntax

```
USETEMPLATESYSTEMUSER TRUE|FALSE
```

Arguments

TRUE

SYSTEM-USER of the USER template record is used for newly added USER record.

FALSE

The SSH user name is used as SYSTEM-USER for newly added USER record.

Default

The default for this parameter is FALSE.

See also

AUTOADDSYSTEMUSERS, AUTOADDSYSTEMUSERSLIKE

Enabling Full TTY Access

SSH2 allows remote SSH clients to establish fully functional OSS shell sessions. This includes the allocation of pseudo terminals (PTYs), which allow remote users to execute full screen applications, such as vi or Emacs.

PTYs are not natively supported by OSS on the NonStop™ server. To overcome this limitation, SSH2 comes bundled with a component named STN. The STN component is also used in another comForte product, SecurTN.

For each PTY allocation request received over SSH, STN will create a dynamic "window" subdevice. STN can also display a service menu to 6530 clients connecting over SSH, allowing users to connect to a service mapped to pre-configured static windows or to a service program started on the dynamic window. This feature allows a complete migration of an existing Telnet access configuration to SSH. Please refer to "Enabling 6530 Terminal Access" in this chapter and to chapter "STN Reference" for further details.

To Start the STN Pseudo Terminal Server Included with SSH2

Note: For cases in which SSH2 was delivered with HP NonStop SSH as part of the RVU or as an independent product for G-Series prior to G06.32, an STN PTY server will be pre-installed as a generic process: SSH-ZPTY ($ZPTY).

1. At the TACL prompt, issue the following commands:

```
CLEAR ALL PARAM
PARAM BACKUPCPU ANY
RUN STN/NAME $PTY, NOWAIT/
```

2. Verify if the process started successfully by checking its status and EMS for any error messages.

Note: For productive use of the STN component, we recommend that you install the EMS template file named “ZSTNTMPL” using standard installation procedures. This will ensure that STN EMS messages will be displayed correctly.

Enabling 6530 Terminal Access

The STN PTY server also supports 6530 pseudo terminals. This enables products such as comForte's MR-Win6530 to create fully functional 6530 terminal sessions with clients over the SSH protocol. 6530 block mode applications, such as ViewPT and Tedit are also supported.

6530 client access can be controlled by setting following attributes of the USER entity of the SSHCTL database:

- ALLOW-CI
- CI-PROGRAM
- CI-COMMAND

By default, SSH2 will start a TACL process on the 6530 PTY device. The TACL will be logged in under the SYSTEM-USER configured for the USER entity. The following sections explain how to configure an alternate command interpreter, and how to enable a service menu similar to TELSERV.

Note: Basic 6530 PTY access requires STN A66 or later.

Configuring an Alternate Command Interpreter

TACL is the default command interpreter that SSH2 starts on a 6530 pseudo terminal. You can use the CI-PROGRAM and CI-COMMAND attributes to assign a different program as the 6530 command interpreter. For example, you can use PATHCOM to run a PATHWAY PROGRAM directly on the pseudo 6530 terminal.

The following SSHCOM commands show how to assign a PATHWAY PROGRAM as the initial program on a 6530 pseudo terminal:

```
>RUN SSHCOM $SSH01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ALTER USER PW.USER, CI-PROGRAM $SYSTEM.SYSTEM.PATHCOM, &
% CI-COMMAND "$PMON; RUN PROGRAM LOGON-PROG"
OK, user PW.USER altered.
%
```

Configuring a Service Menu

STN can also display a service menu to 6530 clients connecting over SSH, allowing users to access a service mapped to pre-configured static windows or to a service program started on the dynamic window. This feature allows the complete migration of an existing Telnet access configuration to SSH.

The following SSHCOM commands show how the STN service menu can be enabled for 6530 pseudo terminals:

```
>RUN SSHCOM $SSH01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ALTER USER SERVICE.USER, CI-PROGRAM *MENU*
OK, user SERVICE.USER altered.
%
```

For non-6530 pseudo terminals the STN service menu can be enabled via:

```
>RUN SSHCOM $SSH01
T9000B03_02DEC2009_SSHCOM
OPEN $ssh01
% ALTER USER SERVICE.USER, SHELL-PROGRAM *MENU*
OK, user SERVICE.USER altered.
%
```

Unless configured otherwise, STN will present TACL as the only available service. Additional services can be added with STNCOM, using the ADD SERVICE and ADD WINDOW commands. Please refer to the "STNCOM Commands" section for further details.

Configuring an STN Service or Window

A user can be enforced to use a pre-configured STN service or window. In this case STN will not display a service menu but will directly give the user access to the pre-configured service or window. This feature allows pre-selection of items defined in the STN service menu depending on the SSH user.

The following SSHCOM commands show how an STN service or window can be enabled for 6530 pseudo terminals:

```
>RUN SSHCOM $SSH01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ALTER USER SERVICE.USER, CI-PROGRAM *MENU* srvc1
OK, user SERVICE.USER altered.
% ALTER USER WINDOW.USER, CI-PROGRAM *MENU* #win1
OK, user WINDOW.USER altered.
%
```

For non-6530 pseudo terminals the STN service or window can be enabled via:

```
>RUN SSHCOM $SSH01
```

```
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ALTER USER SERVICE.USER, SHELL-PROGRAM *MENU* srvc1
OK, user SERVICE.USER altered.
% ALTER USER WINDOW.USER, SHELL-PROGRAM *MENU* #win1
OK, user WINDOW.USER altered.
%
```

The pre-selected service or window ('srvc1' and '#win1' in the examples above) must exist in the STN configuration.

STN services and windows can be added with STNCOM, using the ADD SERVICE and ADD WINDOW commands. Please refer to the "STNCOM Commands" section for further details.

Forcing TACL Access via Server-side Configuration

Usually a remote user can select if the ssh client gets access to an OSS shell or a TACL. In case the user executes a SHELL request e.g.:

```
ssh usr@host
```

and the terminal type is TN6530, then a TACL is created. Also, if the user executes a SUBSYSTEM request with subsystem name tacl, e.g.:

```
ssh –s usr@host tacl
```

then a TACL is started. If the user executes a SHELL request like

```
ssh usr@host
```

and the terminal type is not TN6530, then a shell is started. In case the user starts an EXEC request specifying a command like in:

```
ssh usr@host ls –l
```

then the command is executed in a shell. If a TACL command should be executed, then the gtacl shell command can be used, e.g.

```
ssh usr@host gtacl -c fileinfo
```

or the command tacl with options –c like

```
ssh usr@host tacl –c fileinfo
```

A program can be started in the TACL environment using option –p, e.g.:

```
ssh usr@host tacl –p fup
```

A way to force a user to connect to a TACL is to define an STN service and configure the SSH USER record to use this service.

Assuming a service TACL1 is defined via STNCOM like:

```
ADD SERVICE TACL1, TYPE DYNAMIC, PROG $system.system.tacl, LOGON REQ
```

And the SSH user is configured using SSHCOM commands:

```
ALTER USER usr, SHELL-PROGRAM *MENU* TACL1 FORCE
```

Then both SHELL and EXEC requests, independent of the terminal type will start a TACL.

If the user was successfully authenticated via a different ssh authentication method than none, i.e. the USER attribute ALLOWED-AUTHENTICATIONS was not set to (none), the TACL starts already logged on as user usr because the service was added with “LOGON REQ”.

Using TELSERV as Service Provider

6530 shell channels can also be forwarded to a TELSERV process. This enables a fast and easy migration of an existing complex TELSERV environment to SSH, such as an environment with static windows. To forward 6530 shell requests to TELSERV, specify the CI-PROGRAM as follows:

```
>SSHCOM <ssh2 process name>
%ALTER USER telnetuser, CI-PROGRAM telnet
```

This assumes that TELSERV is listening on port 23 for the same TCPIP process as SSH2. To forward shell requests to a TELSERV listening on a different port or address, specify CI-PROGRAM as follows:

```
%ALTER USER telnetuser, CI-PROGRAM "telnet 192.2.3.4 4023"
```

Similarly, the SHELL-PROGRAM attribute can be prepared as follows (an example using an IPv6 address):

```
ALTER USER test, SHELL-PROGRAM "telnet fe80::a00:8eff:fe02:69d9 5023"
```

6530 shell users (e.g. when connecting a 6530 session over the MR-Win6530 SSH interface) will see the standard TELSERV service menu after the connection is established.

Note: Although TELNET is specified as CI-PROGRAM, SSH2 will not invoke the TELNET program on a STN 6530 pseudo terminal. To provide optimal performance, SSH2 will directly establish a socket connection to the target TELSERV process, which will provide the 6530 terminal device for the session.

Granting Access without SSH Authentication

Under certain circumstances, it is desirable to grant access to specific services without forcing the remote SSH user to authenticate. For example, some services being delivered via SSH may perform their own user authentication. To avoid making users have to enter their credentials twice, the authentication usually performed over the SSH protocol can be turned off. Even without SSH authentication, the connection is still encrypted, protecting any passwords and data transmitted during the service's execution.

CAUTION: When granting unauthenticated SSH access to a resource that performs its own authentication, the user's privileges should be properly locked to prevent unauthorized access to any other resources.

For access without authentication, the SSH2 SERVER can be configured so the authentication method "none" is an ALLOWED-AUTHENTICATION for a user.

The following SSHCOM commands show how to set up a logical user who only authenticates through the SAFEGUARD LOGON program:

```
>RUN SSHCOM $SSH01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ADD USER safeguarduser, ALLOWED-AUTHENTICATION (none), &
% SYSTEM-USER *none*, CI-PROGRAM $SYSTEM.SYSTEM.LOGON, &
% ALLOW-SHELL NO, ALLOWED-SUBSYSTEMS (), ALLOW-TCP-FORWARDING NO
OK, user safeguarduser added.
%
```

In the example above, "safeguarduser" does not require an individual SSH authentication. In this case, the user name serves as a logical service that provides system access via the SAFEGUARD logon program. This service can be shared by multiple individual users. After the session is established, the SAFEGUARD logon program performs user authentication.

Please note that additional attributes limit the access rights of the user to the SAFEGUARD logon program only.

The following SSHCOM commands show how to set up a logical user who is only authenticated with the services started by the STN PTY server:

```
>RUN SSHCOM $SSH01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ADD USER serviceuser, ALLOWED-AUTHENTICATION (none), &
% SYSTEM-USER *NONE*, CI-PROGRAM *MENU*, &
% ALLOW-SHELL NO, ALLOWED-SUBSYTEMS (), ALLOW-TCP-FORWARDING NO
OK, user serviceuser added.
%
```

In the above example, "serviceuser" does not require an individual SSH authentication. Hence, this user represents a logical service that accesses the system via the STN service menu. This service can be shared by multiple individual users. In this scenario, actual user authentication should be performed by STN services.

Again, additional attributes limit the access rights of the user to the STN service menu only.

Single Sign-on with GSSAPI Authentication

Overview

GSSAPI (Generic Security Service Application Programming Interface) is a standardized function interface that provides security services for applications in a mechanism-independent way. In addition, GSSAPI GSSAPI is also a standardized, RFC 4462-compliant way to establish a security context for user authentication and key exchange between an SSH client and server. The prevalent security mechanism supported for use with GSSAPI is Kerberos.

SSH2 supports the RFC 4462 standard for **GSSAPI user authentication** with Kerberos as the security mechanism, both in DAEMON and CLIENT mode. This approach can be used to implement Kerberos-based single sign-on for users connecting with a GSSAPI/Kerberos-enabled SSH client. Since Microsoft Active Directory supports Kerberos, Windows domain users can be enabled to log onto HP NonStop™ Servers without being prompted for a password. If credential forwarding (also known as TGT forwarding) was selected for the session, subsequent SSH connections from the NonStop host to other network resources participating in Kerberos single-sign on can also be accessed without additional authentication.

SSH2 also supports the RFC 4462 standard for **GSSAPI key exchange**, with Kerberos as the security mechanism. This includes the server authentication of the SSH2 daemon via GSSAPI/Kerberos – rather than using its public key, which eliminates the need to manage SSH host public keys on the client side.

Prerequisites

For GSSAPI authentication to work, SSH2 requires a Kerberos package to be installed and properly configured on the same NonStop server. The GSSAUTH server process (which is part of the Kerberos installation) must be running to allow SSH to interface with GSSAPI/Kerberos functionality.

On the remote side, an SSH client or daemon that supports Kerberos authentication via GSSAPI is required. Available options include comForte's MR-Win6530 or J6530 terminal emulator packages, CrystalPoint's OutsideView, Cail's CTT, SSH Tectia, OpenSSH, or a Kerberos-compliant version of PuTTY.

Configuration of the GSSAPI Interface Process

To enable GSSAPI authentication, SSH2 must be configured to locate the GSSAPI authentication interface process (GSSAUTH) of the Kerberos installation. This can be done by specifying the GSSAUTH parameter in the SSH2 startup configuration, for example:

```
RUN SSH2 /NAME $SSH01, ... / ALL; GSSAUTH $GSS; ...
```

Make sure that the GSSAUTH parameter specifies the same process name as that configured for the GSSAUTH process in your Kerberos installation.

Enabling GSSAPI Authentication for a User Account

As any other authentication method, GSSAPI authentication can be enabled or disabled on a per user basis.

The following SSHCOM command illustrates how GSSAPI authentication can be added to the list of allowed authentication methods for a user:

```
>RUN SSHCOM $SSH01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ALTER USER SUPER.OPERATOR, ALLOWED-AUTHENTICATIONS (gssapi-with-mic,password)
OK, user SUPER.OPERATOR altered.
%
```

Note: “gssapi-with-mic” is the standard name in RFC 4462 for GSSAPI-based user authentication. Including “gssapi-with-mic” in the list of allowed authentications will also enable GSSAPI-based key exchange and the “gssapi-keyex” user authentication method. “gssapi-keyex” is a variant of “gssapi-with-mic” that reuses the security context established during GSSAPI key exchange.

GSSAPI authentication can be automatically enabled for newly added users, either by using the SSH2 ALLOWED-AUTHENTICATIONS configuration parameter or by enabling gssapi-with-mic in the ALLOWED-AUTHENTICATIONS attribute of a user that has been configured with the SSH2 AUTOADDSYSTEMUSERSLIKE parameter.

Authorizing Kerberos Principals for Logon

For customers using a Kerberos solution, Kerberos authentication via GSSAPI allows the SSH2 daemon to securely identify the user’s Kerberos principal name (such as the Microsoft Active Directory user ID). Using this unique Kerberos identity, users can be authorized to access one or more NonStop user accounts.

The authorization can be controlled either implicitly or explicitly, as described in the following sections.

Implicit Authorization

Implicit authorization takes advantage of the Kerberos default authorization rule:

If host H is in the realm R, the Kerberos principal u@R is allowed access to the account u@H.

This rule means that a Kerberos principal can access an SSH user account, if the user name exactly matches the user portion of the Kerberos principal name, and the local NonStop host is in the same realm. For example, if the NonStop server is configured in a Microsoft Active Directory, an Active Directory user may access an SSH account with a matching user name.

For example, if the NonStop host is configured as NonStop@COMPANY.COM, a user JohnSmith@COMPANY.COM can be implicitly authorized to logon as SUPER.OPERATOR as follows:

```
>RUN SSHCOM $SSH01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% ADD USER JohnSmith, SYSTEM-USER SUPER.OPERATOR, ...
OK, user JohnSmith added.
%
```

Another implicit authorization method would be to create a Safeguard ALIAS:

```
>SAFECOM
SAFEGUARD COMMAND INTERPRETER - T9750H04 - (13AUG2008)    SYSTEM  \NONSTOP
= ADD ALIAS JohnSmith, SYSTEM-USER SUPER.OPERATOR, ...
OK, user JohnSmith added.
%
```

If the SSH2 AUTOADDSYSTEMUSER option is disabled, the ALIAS must also be added to the NonStop SSH database using the SSHCOM ADD USER command. Otherwise, if the SSH2 AUTOADDSYSTEMUSER option is TRUE and gssapi-with-mic is enabled for automatically added users, then creating a Safeguard ALIAS for the Kerberos user principal will be sufficient to grant SSO access.

Explicit Authorization

Explicit authorization involves defining an access control list containing specific Kerberos principals authorized to access an account. The access control list can be defined using the SSHCOM USER PRINCIPAL attribute.

For example, if the NonStop host is configured as NonStop@COMPANY.COM, a user JohnSmith@COMPANY.COM can be explicitly authorized to logon as SUPER.OPERATOR as follows:

```
% ALTER USER SUPER.OPERATOR, PRINCIPAL JohnSmith@COMPANY.COM
OK, user SUPER.OPERATOR altered.
%
```

Note: You can authorize multiple Kerberos principals to logon as a specific NonStop user by specifying multiple PRINCIPAL attributes in one or more ALTER USER commands. HP does not currently offer a Kerberos solution, but such a solution can be purchased from an HP NonStop partner and applied to your system.

Restricting Incoming and Outgoing Connections

Port forwarding on a global level is determined by the SSH2 parameter ALLOWTCPFORWARDING. The user attribute ALLOW-TCP-FORWARDING is used to grant or deny port forwarding on a user level.

Sometimes a finer granularity is needed to restrict forwarding to specific hosts. The RESTRICTION-PROFILE objects and the user attribute ALLOW-GATEWAY-PORTS can be used to configure forwarding restrictions with more granularity.

Rejecting Gateway Ports

If a user specifies the "–g" SSH2 option when initiating a port forwarding request, the listening on the local port will not occur on the loopback IP address 127.0.0.1 (localhost) but on all subnets defined for the TCP/IP process. Such a port is called a gateway port as the host can be used as a gateway to a third host. A port forwarding request will be denied if the value of the user attribute ALLOW-GATEWAY-PORTS is set to FALSE. The user can still open non-gateway ports listening on 127.0.0.1.

Restricting External Access to SSH2 Process

The restriction profile attribute CONNECT-FROM can be used in environments in which some remote hosts should not be allowed to connect to a specific SSH2 instance running on a NonStop server. The value is a list of host names and IP addresses or patterns that are allowed to connect to the port SSH2 is listening to for SSH requests (default: 22).

The SSH user specified in the incoming SSH request is checked against the corresponding user record in SSHCTL. The user attribute RESTRICTION-PROFILE is used to access the RESTRICTION-PROFILE object, which contains the setting for CONNECT-FROM. If a RESTRICTION-PROFILE object and a CONNECT-FROM value is configured, the host/IP address of the incoming SSH connection request will be checked against the list of hosts/IP addresses defined in CONNECT-FROM. The incoming SSH2 request is accepted only if a match is found, otherwise it is rejected.

Restricting Internal Access to Remote SSH2 Hosts

If a user should not be allowed to connect to all available remote SSH instances, the SSH2 user configuration can be used to restrict outgoing access via the RESTRICTION-PROFILE attribute CONNECT-TO. The CONNECT-TO attribute defines a list of host/port combinations that a user is allowed to reach via a specific SSH2 instance. No pattern matching is allowed but several hosts can be defined and several ports can be specified per host.

If the user attribute RESTRICTION-PROFILE is defined and the CONNECT-TO attribute of the restriction profile is set, the SSH2 process limits access to the configured host/port combinations only when starting an outgoing connection for that user.

Restricting Local Ports used for Port Forwarding

In an environment in which some users should not be allowed to listen on any (unused) local ports for forwarding purposes, a list of allowed 0.0.0.0/port and 127.0.0.1/port combinations can be defined. The RESTRICTION-PROFILE attribute PERMIT-LISTEN holds this list.

For remote clients, the user specified in the incoming SSH request is checked against SSHCTL.

This forwarding listen port restriction is applied if the attribute RESTRICTION-PROFILE of the user record is set and the PERMIT-LISTEN attribute of the corresponding restriction profile record is configured.

Restricting Remote Hosts/Ports for Port Forwarding

If a user should not be permitted to open a tunnel to any host/port for forwarding purposes, administrators can configure specific host/port combinations for specific users. Host/port combinations can be specified via the RESTRICTION-PROFILE attribute PERMIT-OPEN, which corresponds to the OpenSSH "permitopen=" option.

For remote clients, the user specified in the incoming SSH request is checked against SSHCTL.

This forwarding restriction is applied if the attribute RESTRICTION-PROFILE is set in the user record and the PERMIT-OPEN attribute is configured in the corresponding restriction profile.

Restricting access to forwarding tunnels

In scenarios in which a user is allowed to create a forwarding tunnel, administrators can require the definition of which hosts have access to the tunnel. Using the RESTRICTION-PROFILE attribute FORWARD-FROM, a list of hosts/IP addresses/patterns can be defined that identify those hosts that are allowed to use a tunnel created by a specific user. In this case, the list of allowed hosts is determined by the user who opened the tunnel, if configured accordingly.

For remote clients the user specified in the incoming SSH request is checked against SSHCTL.

This forwarding-from restriction is applied if the RESTRICTION-PROFILE attribute of the user record is set and the FORWARD-FROM attribute of the corresponding restriction profile record is configured.

Load Balancing

With SSH2, it is possible to distribute the CPU load generated by the encryption of SSH sessions across multiple processors of a NonStop system. This is true for both inbound and outbound sessions.

Load-Balancing Outbound SSH Sessions

For outbound sessions, CPU load balancing can be achieved by starting multiple SSH2 instances and distributing client processes across processors. The load-balancing for outbound ssh sessions depends on client processing and can only be influenced by settings in the client environment controlling the client's processing.

All clients delivered with SSH2 (SSH, SSHOSS, SFTP, and SFTPOSS) employ a heuristic method in which an SSH2 process is opened to create the outbound session. The heuristic method works as follows:

1. If no explicit SSH2 process is configured (which is done by specifying the –S option on the command line), the client evaluates first the define =SSH2^PROCESS^NAME and then the environment variable SSH2_PROCESS_NAME to determine the process name of the SSH2 instance to connect to.
2. If neither define =SSH2^PROCESS^NAME nor environment parameter SSH2_PROCESS_NAME exists, the client evaluates an environment variable named SSH2PREFIX to determine the process name prefix of the SSH2 instances. The default is "$SSH".
3. If an open action fails, the client will look for an instance of an SSH2 process with the next higher processor number, up to 15. After processor number 15 is searched, "00" will be tried. For example, if the SSH2PREFIX is set to $ABC and there are two SSH2 processes running, one in cpu 4 with port 22, subnet $ztc0, and name $ABC04, and one in cpu 5 with port 22, subnet $ztc1, and name $ABC05, an invocation of client SSH with no -S and -p params connecting to a remote Unix box will find one of the two SSH2 processes, depending in which cpu the client SSH was started: $ABC04 if SSH was started in a cpu other than 5, and $ABC05 if it was started in cpu 5.
4. If all process names fail, the client will terminate with an error message.

The process names of the SSH2 instances serving the clients must be correctly configured to facilitate this heuristic method. For example, you could decide to start an SSH2 instance in every CPU of your system, naming the instances according to the number of the CPU they are running in:

```
RUN SSH2/NAME $SSH00, CPU 0, …/ …
RUN SSH2/NAME $SSH01, CPU 1, …/ …
…
```

After you have started multiple SSH2 instances in the manner described above, the distribution of the client processes over CPUs will also ensure that the sessions are distributed across the available SSH2 instances. This distribution of client processes can either be achieved manually, or by using any standard load-distributor tool available on your system.

Load-Balancing Inbound SSH Sessions

For incoming sessions, SSH2 can facilitate the round-robin filtering feature of TCPIPv6. In addition, parallel round-robin filtering allows you to start multiple SSH2 listening processes in different processors that share the same port.

To enable round-robin filtering with SSH2, you have to configure the PTCPIPFILTERKEY parameter for every SSH2 instance listening on the same port as follows:

```
RUN SSH2/NAME $SSH00, CPU 0, .../ ALL; PORT 22, PTCPIPFILTERKEY mykey
RUN SSH2/NAME $SSH01, CPU 1, .../ ALL; PORT 22, PTCPIPFILTERKEY mykey
```

After you have started multiple SSH2 processes in the manner described above, inbound SSH sessions will then be distributed across the SSH2 instances in a round-robin manner.

The application processes started by SSH2 for incoming connection can be distributed over CPUs on a user level via different settings of USER attribute CPU-SET and SFTP-CPU-SET. The SSH2 parameters CPUSET and SFTPCPUSET allow defining default values for these USER attributes on a global level. If multiple CPUs are configured, then these will be used in a round-robin fashion.

Another way of load balancing of incoming SSH connections is to configure multiple IP processes for one SSH2 process (see parameter SUBNET) and let users connect to different IP addresses of the NonStop system. In this way the TCP/IP traffic load is distributed over the CPUs if the configured TCP/IP processes run in different CPUs.

Fault Tolerance

SSH2 can be configured to ensure constant availability of NonStop-based SSH applications across the network. Running on the Guardian platform, SSH2 takes advantage of the fundamental availability characteristics of NonStop™.

SSH2 services can be configured as generic processes, enabling automatic recovery from failures, such as CPU outages. SSH2 can also be started as a NonStop process pair. Both mechanisms will not prevent sessions to fail after the primary CPU of the SSH2 process goes down. However, SSH2 will restart operation in a backup CPU, ensuring that clients can reconnect immediately.

Configuring SSH2 as a NonStop Process Pair

SSH2 can easily be started as a NonStop process pair by specifying the BACKUPCPU parameter as follows:

```
RUN SSH2/ NAME $SSH00, CPU 0, .../ ALL; BACKUPCPU ANY; ...
```

In case of a failure of the primary CPU, the backup process of SSH2 will take over and restart the operation.

Configuring SSH2 as a Generic Process

The following sample SCF commands can be used to configure a SSH2 server as a generic process:

```
ALLOW ALL ERRORS
ASSUME PROCESS $ZZKRN

ABORT #SSH2
DELETE #SSH2

ADD #SSH2, AUTORESTART 10,                                      &
  HOMETERM $ZHOME,                                              &
  PRIORITY 158,                                                 &
  PROGRAM $SYSTEM.COMFSSH2.SSH2,                                &
  DEFAULTVOL $SYSTEM.COMFSSH2,                                  &
  NAME $SSH2,                                                   &
  STARTUPMSG "SERVER; PORT 22; SUBNET $ZTC01; LOGCONSOLE *;     &
    LOGFILE SSHLOG ",                                           &
  STARTMODE MANUAL,                                             &
  USERID SUPER.SUPER ,                                          &
  CPU FIRST

START #SSH2
INFO #SSH2
STATUS #SSH2
```

Before running SSH2 as a generic process, we recommend that you have a working RUN SSH2 command at the TACL level. This command should be easy to convert to the respective SCF ADD command. For example, the SSH2 startup line parameters are specified with the STARTUPMESSAGE parameter.

If running SSH2 as a generic process, we recommend that users send the SSH2 log output to a log file instead of writing it to the home terminal, which is the default approach. In the example above, console logging is turned off, while log messages are written to the SSHLOG file on the default volume.

If you want to configure multiple SSH2 servers listening on the same port with parallel library TCP/IP or TCP/IPV6 round-robin filtering, you may specify the filter key with the PTCPIPFILTERKEY configuration parameter or add define =PTCPIP^FILTER^KEY for the generic process (defines can be added to generic processes since G06.28/H06.06).

Likewise, you can use the TCPIPHOSTFILE, TCPIPNODEFILE, and TCPIPRESOLVERNAME parameters to configure TCPIP settings or the corresponding DEFINEs.

Please refer to the *SCF Reference Manual for the Kernel Subsystem* in the HP NonStop™ documentation set for further details.

Choosing a Persistence Mechanism

Determining whether it is more effective to configure SSH2 as a NonStop process pair or as a generic process depends on your system environment and the expected SSH transfer volume.

For an environment with low volumes of SSH traffic, it may be sufficient to run a single SSH2 process pair. However, if you expect a higher traffic volume, you may want to distribute the CPU load across the available CPUs on your system. This can be done by starting multiple SSH2 instances as described in the "Load Balancing" section above. Running multiple SSH2 instances may have an influence on the fault-tolerance mechanism you choose. Following are key considerations:

- When running multiple process pairs of SSH2 listening on the same port, you should not start a primary SSH2 process in a CPU that is used as a backup process by another SSH process pair. If you do, there will be a conflict with two processes trying to listen on the same port in case of failover. Consequently, the maximum number of SSH2 process pairs listening on the same port is the number of CPUs on your system divided by two. Furthermore, the CPU load generated by the SSH encryption would only be distributed across the primary CPUs of the SSH2 instances.
- When running SSH2 as a generic process, you can rely on the persistence manager to restart SSH2. It is not necessary to start SSH2 as a process pair. Hence, if you want to distribute the load evenly across all processors, it may be better to configure a generic SSH process in each CPU that would be restarted automatically when a CPU comes up after a failure.

Processing of DEFINEs

SSH2 has been enhanced to propagate almost all defines found in the SSH2 process context to TACL and shell processes started by SSH2 directly. Exceptions are:

The =_DEFAULTS DEFINE is set from the Guardian user configuration.

In case parameters PTCPIPFILTERKEY, TCPIPHOSTFILE, TCPIPNODEFILE or TCPIPRESOLVERNAME were specified the corresponding defines propagated contain the values taken from these parameters, i.e. the defines in SSH2 process context will be overwritten.

If define =TCPIP^PROCESS^NAME exists in the process context it will be propagated and the SUBNET parameter value will be ignored (see parameter SUBNET). If define =TCPIP^PROCESS^NAME does not exist in the process context the SUBNET parameter value will be used to create a define =TCPIP^PROCESS^NAME and it will be propagated to newly started TACL and shell processes.

If define =CIP^COMPAT^ERROR exists in the SSH2 process context it will be propagated and the CIPCOMPATERROR parameter value will be ignored (see parameter CIPCOMPATERROR). If define =CIP^COMPAT^ERROR does not exist in the process context a CIPCOMPATERROR parameter value other than '*' will be used to create a define =CIP^COMPAT^ERROR and it will be propagated to newly started processes.

The processing of TCP/IP related defines and corresponding parameters is limited to creation/overwriting of defines. If neither of the SSH2 TCP/IP parameters are set, then the existing TCP/IP defines/parameters determine the processing. The actual processing is solely done in the TCP/IP runtime libraries, i.e. if the relevant TCP/IP parameters like =TCPIP^RESOLVER^ORDER and TCP/IP related defines are set, then the resolver order should be as configured.

There is a special processing the SSH2 process executes regarding name resolving during startup: Without explicit settings the TCP/IP stack uses DNS for name resolving. This causes long delays if name resolving is incorrectly configured. If a name resolving test at startup takes too long, then the SSH2 process assumes the name resolving is not correctly configured and the define =TCPIP^HOST^FILE is set to the default value. A warning is logged in this case ("Disabling incorrectly configured DNS resolving").

A new define =SSH2^PROCESS^NAME will be created and propagated. It contains the name of the SSH2 process, which started the TACL or shell process. The SSH clients (objects SSH, SSHOSS, SFTP and SFTPOSS) make use of this define to look up the SSH2 server process before the CPU dependent lookup using SSH2PREFIX is tried. Those SSH clients running within a shell started by an SSH2 server process no longer require specifying the SSH2 server process via the -S flag.

Defines may have unwanted influence on the processing of started processes, e.g. if a TCP/IP application is started that needs to use different DEFINE settings.

If defines should not be forwarded to processes started by the SSH2 process, then parameter PROPAGATEDEFINES can be set to FALSE and the forwarding of defines will be suppressed (default is TRUE). The define =_DEFAULTS is always propagated to new processes, independent of the setting for SSH2 parameter PROPAGATEDEFINES.

Setting of PARAMs

SSH2 may create the following PARAMs when starting a TACL:

SSH-ORIGINAL-COMMAND

The command that was specified in an exec request. This can be different to the actually executed command, in case a “forced command” is defined (USER attribute CI-COMMAND).

Setting of Environment Variables

SSH2 creates the following environment variables when starting a shell:

SSH_CONNECTION

This environment variable contains host and port information, each separated by a space character:

```
<remote address> <remote port> <local address> <local port>
```

Example:

```
SSH_CONNECTION=10.0.0.12 40719 10.0.0.196 22
```

SSH_CLIENT

This environment variable contains remote host/port and local port information, each separated by a space character:

```
<remote address> <remote port> <local port>
```

Example:

```
SSH_CLIENT=10.0.0.12 40719 22
```

TERM

This environment variable holds the terminal type.

Example:

```
TERM=xterm
```

LOGNAME

The user name as received from a remote client (the name of a user defined in SSHCTL).

Example:

```
LOGNAME=test.us
LOGNAME=mike
```

SSH_TTY

The pseudo terminal allocated for the session.

Example:

```
SSH_TTY=/G/pty35/#zwn0001
```

SSH2_PROCESS_NAME

The SSH2 process that started the shell process.

Example:

```
SSH2_PROCESS_NAME=$SSH35
```

HOME

The shell home directory of the user.

Example:

```
HOME=/home/test
```

SSH_ORIGINAL_COMMAND

The command that was specified in an exec request. This can be different to the actually executed command, in case a "forced command" is defined (USER attribute SHELL-COMMAND).

Example:

```
SSH_ORIGINAL_COMMAND=ls -l
```

ENV

Value taken from USER attribute SHELL-ENVIRONMENT

Examples:

```
ENV=$HOME/setenvvars
ENV=/etc/nonloginshellenvs
ENV=~/testenv
```

TCP/IPv6 Configuration

The IPv6 standard differs from the IPv4 standard in many ways. The TCP/IP configuration for IPv4 and IPv6 on NonStop servers is different in several aspects as well, see documents and links listed in section "Related Reading".

But from NonStop SSH and comForte SecurSH/SecurFTP product's standpoint the differences are mainly related to the new address formats of IPv6, new defines and different modes the NonStop TCP/IP processes with IPv6 support can run in.

IPv6 Address Formats

IPv4 uses 32 bits for an Internet Protocol address, and can therefore support 2^{32} (4,294,967,296) addresses. IPv6 uses 128-bit addresses, i.e. the new address space supports 2^{128} ($3.4x10^{38}$) addresses.

Although IPv4 addresses may be presented in various hexadecimal, octal, or binary representations, they are canonically represented in dotted decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots, e.g., 172.1.2.3. Each decimal number represents 8 bits (one octet) of the IPv4 address.

IPv6 addresses are not only longer than IPv4 addresses but there can be several valid representations of an IPv6 address. An IPv6 address is represented as eight groups of four hexadecimal digits separated by colons, e.g. 2001:0db8:0000:0000:1319:0000:0000:7344. Each group represents 16 bits (two octets) of the IPv6 address. Leading zeros are usually dropped, resulting in the valid representation 2001:0db8:0:0:1319:0:0:7344. Further simplifying (RFC 4291) allows to replace a sequence of 0 groups to one "::" group, resulting in 2001:0db8::1319:0:0:7344 (a maximum of one "::" sequence is allowed). The original example address can also represented as 2001:0db8:0:0:1319::7344. Usually the longest sequence of zero groups is replaced by "::". If there is more than one sequence of 0 groups of the same length, the first sequence is replaced by "::".

Another IPv6 representation uses dotted decimals for the last 4 octets of an IPv6 address, especially used for IPv4 compatible IPv6 addresses like ::13.1.68.3 and IPv4-Mapped IPv6 addresses like ::FFFF:129.144.52.38.

In cases where a numeric element like a port (or any or hexadecimal element not belonging to the IP address) is appended to an IP address separated by a colon, the IP address must be enclosed with square brackets if the IP address is an IPv6 address, e.g. [2001:0db8::1319:0:0:7344]:4567. Otherwise the port could be misinterpreted as part of the address (2001:0db8::1319:0:0:7344:4567 is a valid IPv6 address).

The representation for the unspecified address in IPv4 is "0.0.0.0". The unspecified address in IPv6 (sequence of zero groups) can be represented as "::" or "0::0" (other forms are valid as well). The SSH2 process usually uses "0::0" as representation of the unspecified IPv6 address but accepts any other representation as well.

All the listed variants of IPv6 address representation are supported by SSH2.

Usage of IPv6 Addresses

Representations of IPv6 addresses are used for restricting the listening (see SSH2 parameters INTERFACE), for defining the local IP address when outgoing connections are established (SSH2 parameter INTERFACEOUT, ssh/sftp client option –oBindAddress). Also, IPv6 address representations can be used instead of host names mapping to IPv6 addresses when specifying the target host for ssh and sftp clients.

In addition, IPv6 addresses are used in all places where only IPv4 addresses could occur in pre-0092 releases (square brackets may be needed for IPv6 addresses if required). This not only includes database entries, SSHCOM commands, output of SSHCOM commands but log messages and audit messages as well.

Database entities that can hold IPv6 addresses:

Entity USER fields:

- LAST-IP-ADDRESS
- CI-PROGRAM (e.g. when configured with "TELNET <ip-address> <port>")

Entity RESTRICTION-PROFILE fields:

- CONNECT-FROM
- CONNECT-TO
- PERMIT-LISTEN
- PERMIT-OPEN
- FORWARD-FROM

Entity KNOWNHOST fields:

- Name (identifier) of a KNOWNHOST record
- ADDRESSES

Entity PASSWORD fields:

- Name (identifier) of a PASSWORD record

IP Mode

Similar to the FAMILY configuration of TCP/IP monitor process and subnets, the SSH2 process supports control over the IP mode the SSH2 process is running in. A new SSH2 parameter IPMODE has been added.

The SSH2 parameter IPMODE allows restricting communication to IPv4 or IPv6 or allowing both types. The accepted values for parameter IPMODE are:

- IPV4 – allows IPv4 communication only (can be used when accessing a TCP/IP process running object TCPIP or a TCPIP process running TCP6SAM/CIPSAM with a monitor process configured with FAMILY INET or DUAL).
- IPV6 – allows IPv6 communication only (can be used when accessing a TCP/IP process running object TCP6SAM/CIPSAM with a monitor process configured with FAMILY INET6 or DUAL
- DUAL – allows both IPv4 and IPv6 communication (can be used when accessing a TCP/IP process running object TCP6SAM/CIPSAM with a monitor process configured with FAMILY INET, INET6 or DUAL).

Generally, an SSH2 process can only support a protocol family if the underlying TCP/IP process provides support for that protocol family. If, for example, SSH2 is configured with IPMODE IPV4 and the TCP/IP process accessed by this SSH2 process is configured with FAMILY INET6, then no communication is possible at all.

TCP/IPv6 Considerations

Using Link Local Addresses for Loopback

While it is possible to use link local addresses within a network segment without problems, there are restrictions using link local addresses for a loopback connection with a TCP/IP CLIM involved. The CIP TCP/IP implementation requires specifying a local TCP/IP address to bind to when trying to establish a loopback connection via CIP TCP/IP. Error 4022 is the result if no specific local IP address is bound in this case.

A local bind address can be specified via the sftp and ssh client option -oBindAddress=<bind-address>, see sections "SSH Client Command Reference" and "SFTP Client Command Reference".

Another way to ensure a local bind address is set depends on the SSH2 parameter INTERFACEOUT: If the value of that parameter is not the any address (0.0.0.0 or 0::0) but a specific IP address valid for the configured SUBNET, then this configured local IP address is bound for every outbound connection.

Alternatively the IPv6 address ::1 can be used as target address without the need for specifying a local bind address.

TCP/IPv6 Migration and Backout

Start Using TCP/IPv6

After the TCP/IP processes have been prepared for IPv6 support the SSH2 processes can be enabled for IPv6 by restarting them with parameter IPMODE set to IPv6 or DUAL. The default for this parameter is value IPv4, i.e. the SSH2 process does not automatically switch to IPv6. This is done because errors would occur when an SSH2 process starts in IPMODE IPv6 or DUAL against a TCP/IP process not supporting IPv6. The object the TCP/IP process is running may not support IPv6 at all ($SYSTEM.SYSnn.TCPIP) or the object may principally support IPv6 but is not configured for IPv6.

As listed in section "Usage of IPv6 Addresses", various SSH database records can contain IPv6 addresses. These fields are updated either when sessions are established (USER field LAST-IP-ADDRESS, name field of KNOWNHOST and PASSWORD entity, ADDRESSES field of KNOWNHOST record) or when the entities are modified via SSHCOM commands (USER field CI-PROGRAM when configured with “TELNET <ip-address> <port>”) and RESTRICTION-PROFILE attributes).

It is recommended to make a copy of each RESTRICTION-PROFILE record before adding any IPv6 addresses/patterns to any of the RESTRICTION-PROFILE records. This can easily be done using SSHCOM command ADD RESTRICTION-PROFILE with LIKE option, e.g.:

```
ADD RESTRICTION-PROFILE ABC_copy, LIKE ABC
```

This step allows a simple way of backing out the IPv6 related changes, in case that is needed.

When multiple SSH2 processes access the same SSH database, then all SSH2 processes should run the same SSH2 object (i.e. either one that supports IPv6 or one that doesn't).

Reverting Back to Pre-IPv6 SSH2 Release

Due to database record versioning there is no change made in the SSH2 database by an SSH2 object with IPv6 support that would cause problems when an SSH2 object without IPv6 support accesses this database. Therefore a backout of an SSH2 IPv6 release to a pre-IPv6 SSH2 release does not represent a problem.

Obviously any change to CI-PROGRAM that was made using format "TELNET <ip-address> <port>" with an IPv6 IP address for the <ip-address> part will no longer work in an IPv4 environment and must be changed back to using an IPv4 address.

Similarly, any changes to RESTRICTION-PROFILE that include IPv6 addresses should be reverted. If a copy of restriction profiles had been made, then simple rename commands will be sufficient:

```
RENAME RESTRICTION-PROFILE <active-profile-name>, <saved-IPv6-profile>
RENAME RESTRICTION-PROFILE <saved-IPv4-profile>, <active-profile-name>
```

For example:

```
RENAME RESTRICTION-PROFILE ABC, ABC_IPV6
RENAME RESTRICTION-PROFILE ABC_copy, ABC
```

If there are RESTRICTION-PROFILE records left containing IPv6 addresses/patterns, then these do not represent a problem: these IPv6 addresses/patterns would just not match when checked against IPv4 addresses being processed by an SSH2 process without IPv6 support.

IPv6 addresses stored in the ADDRESSES field of KNOWNHOST entities will be ignored by SSH2 processes without IPv6 support. A KNOWNHOST entry with an IPv6 address as part of the name cannot be modified or removed using an SSH2 version without IPv6 support but an SSH2 process that supports IPv6 started in ADMIN mode can be used to do that, if required.

A pre-IPv6 SSH2 process builds the key (name of PASSWORD entry) using an IPv4 address and will therefore not find any entries containing IPv6 addresses; that is, no change is required when reverting to a pre-IPv6 SSH2 release. Such

PASSWORD entries cannot be modified or deleted using an SSH2 release without IPv6 support. But again, an SSH2 process that supports IPv6 started in ADMIN mode can be used to do that, if needed.

Multiple IP Process, Multiple IP Address Considerations

Multiple IP Process Configuration

If the define =TCPIP^PROCESS^NAME is used to specify the TCP/IP process SSH2 should use, then it is not possible to configure multiple IP processes. Instead of this define it is required to use parameter SUBNET (and the define must be deleted from the TACL environment before starting the SSH2 process as the define has precedence over parameter SUBNET).

Parameter SUBNET can be a list of IP process names, e.g. $ZTC0,$ZTC1,$ZSAM1,$ZSAM2. Assuming that parameters INTERFACE and INTERFACEOUT are not set (defaulting to the ANY address), SSH2 will start a listener for each of the configured IP processes on the ANY address on the configured port.

Such a configuration can be helpful to simplify the SSH configuration in environments with many TCP/IP processes but little traffic over each IP process.

Multiple Allowed Listen IP Address Configuration

Before the introduction of support for multiple IP processes there has been support for multiple IP addresses. There was just the restriction that all IP addresses had to be configured in one IP process and it was not possible to start a listen on a subnet of configured IP addresses. It had to be either one IP address or all (achieved by using the ANY address for listening).

Now it is possible to listen on a set of IP addresses which can be configured in a set of IP processes. The set of listen IP addresses is specified via parameter INTERFACE and the set of IP processes is configured via parameter SUBNET.

Example:

Assuming INTERFACE is set to 1.2.3.4,1.2.3.5 and SUBNET is configured as $ZTC1, which has configured subnets for 1.2.3.6 in addition to 1.2.3.4 and 1.2.3.5. In this case two listens are initiated against the IP process $ZTC1, one for IP address 1.2.3.4 and one listen against IP address 1.2.3.5.

In a different scenario the address 1.2.3.4 may be configured in process $ZTC1 and 1.2.3.5 in process $ZTC0. Both processes are assumed to have other subnets. With INTERFACE again set to 1.2.3.4,1.2.3.5 and SUBNET set to $ZTC0,$ZTC1 the SSH2 process will again issue two listen operations but this time one for IP address 1.2.3.4 against IP process $ZTC1 and for IP address 1.2.3.5 against IP process $ZTC0.

Should all IP addresses configured in a specific IP process be listed in parameter INTERFACE, then only one listener for the ANY address is started against that IP process and not one for all listed/configured IP addresses of that IP process.

If at least one IP address is listed in the parameter INTERFACE value that is configured in an IP process, then there will be at least one listen started against the IP address. If none of the IP addresses of the INTERFACE value match, then no listener gets started.

If one IP process is configured (via define =TCPIP^PROCESS^NAME or parameter SUBNET), then all IP addresses configured in INTERFACE must correspond to a subnet in the one IP address.

If more than one IP process is configured (via parameter SUBNET), then the values in INTERFACE may belong to any of the configured IP processes. Listeners will only be started for those IP addresses that match a subnet of an IP process. In case none of the INTERFACE values correspond to any of the subnets of an IP process, then no listeners get started for that IP process.

The same IP address may be configured in more than one IP process. If that IP address is configured in INTERFACE, then a listen on such an IP address is issued against each of the configured IP processes.

There may be the requirement to listen on specific IP addresses of some IP processes but to listen on the ANY address for other IP processes. This can be achieved by specifying the ANY address in INTERFACE, in addition to the specific IP addresses.

Example: A listen is required on IP address 1.2.3.4, which is configured in process $ZTC1. Additionally a listen needs to be issued for the ANY address against $ZTC0. Then the parameter INTERFACE would be set to 1.2.3.4,0.0.0.0 and SUBNET value would be $ZTC0,$ZTC1.

Multiple Allowed Bind IP Address Configuration

A specific bind address could be specified from a local SSH[OSS]/SFTP[OSS] client via runtime option -oBindAddress=<bind-address> when INTERFACEOUT was not set (configured with the ANY address). If such option did not exist on the client command line in this case, the actual bind address was determined by the TCP/IP process. An administrator could only select one specific local IP address as local bind address by configuring INTERFACEOUT to that specific IP address. With such a configuration any -oBindAddress options specified on the client command line is ignored and the bind address configured via INTERFACEOUT is used.

With the support of multiple IP addresses for INTERFACEOUT, it is possible to allow a set of IP addresses as bind addresses. If the -oBindAddress option of a client selects one of the IP addresses configured in INTERFACEOUT, then the address supplied from the client will be used as local bind address for the connection.

If the client does not specify a bind address, then the SSH2 process selects one of the configured IP addresses in INTERFACEOUT according to a round-robin algorithm that selects an IP address by first selecting an IP process (should there be more than one IP processes configured in SUBNET) taking the CPU the IP process is running in for the round-robin selection. Then one of the IP addresses of that IP process, which is also listed in INTERFACEOUT is selected. In this way the outgoing connections are distributed over all CPUs the configured IP processes are running in.

Multiple Target IP Address Selection

With DNSMODE set to FIRST or if an IP address is specified for the target host, multiple target IP addresses do not occur. But if parameter DNSMODE is set to ALL and a name is specified as target host, then the host name may get resolved to multiple IP addresses. If that is the case one IP address must be selected for the actual connection. This is done in a round-robin fashion over all target IP addresses a specific SSH2 process has seen in the recent past. This means that the target IP address is selected from the list of resolved IP addresses by checking how often an outgoing connection has been established in the last time interval and picking the IP address with the smallest number of outgoing connections happened during the past interval. Information about connections established before the start of that interval will be dropped.

In this way the outgoing connections are distributed over all IP addresses a specific host name is resolved to.

TACL Subsystem and Command Interpreter Configuration

Enhanced EXEC Processing

The processing of EXEC requests (ssh client started with a remote command on the ssh command line) has been enhanced in version 0097 to add flexibility. It is now possible to let a user execute single TACL commands or TACL macros or a command interpreter other than TACL even though the subsystem TACL is not allowed for the user (ALLOWED-SUBSYSTEMS does not contain tacl).

Previously, the execution of CI-PROGRAM via TACL command on the SSH client command line was rejected if tacl was not an allowed subsystem. Now the tacl subsystem can be removed from the list of ALLOWED-SUBSYSTEMS but the execution of commands via “tacl -c <command>” and “tacl -p <program> <cmd>” is still allowed as long as the USER attribute ALLOW-CI is set to YES.

If an EXEC request is received and subsystem tacl is not allowed, CI-PROGRAM is left at the default value and CI-COMMAND is not configured, then either -p or -c must be specified. Otherwise the user would get a TACL prompt, which should not be allowed if tacl is not an allowed subsystem. The enhanced EXEC processing includes the possibility to use subsystem tacl and CI-PROGRAM independently. Previously the subsystem tacl was initiated for an EXEC tacl request. In order to be compatible with the previous behavior EXEC tacl still starts subsystem tacl if tacl is an allowed subsystem. But now it is possible to specify a new command "ci" (instead of "tacl") on the SSH client command line with options “-c <cmd>” and “-p <program> <cmd>” with the same meaning as the tacl -p and -c options.

The processing of EXEC ci is as follows, if ALLOW-CI is set to YES:

- Command on ssh client command line is "ci":

  The value of USER attribute CI-PROGRAM is started as command interpreter (default: $SYSTEM.SYSTEM.TACL). If additionally CI-COMMAND is configured, then this command is executed. If no command is specified and tacl is not an allowed subsystem, the request will be rejected.

- Command on ssh client command line is "ci -c <cmd>":

  The value of USER attribute CI-PROGRAM is started as command interpreter (default: $SYSTEM.SYSTEM.TACL) and the command <cmd> is executed by the command interpreter unless CI-COMMAND is configured. In this case the command <cmd> is ignored (but available via PARAM SSH-ORIGINAL-COMMAND) and the command configured under user attribute CI-COMMAND is executed.

- Command on ssh client command line is "ci -p <program> [<cmd>]":

  The command interpreter program <program> is started (default subvolume if not specified is $SYSTEM.SYSTEM) and if <cmd> is specified, then this command is executed. If no <cmd> is specified, then the user will get the prompt of the command interpreter and can enter commands interactively.

  It is possible that a user specifies "ci -p tacl" but the access of tacl may not be allowed for the user. Therefore a new USER attribute ALLOW-CI-PROGRAM-OVERRIDE determines if a user is allowed to use "ci -p". The default value for attribute ALLOW-CI-PROGRAM-OVERRIDE is NO.

With this enhancement, if subsystem tacl is not allowed, an EXEC request like "tacl -c <cmd>" or "tacl -p <program> <cmd>" will be automatically converted to "ci -c <cmd>" and "ci -p <program> <cmd>", respectively, and handled accordingly. In any case, if subsystem tacl is not allowed, then a user will not get a tacl prompt.

Default configuration

The default configuration allows for subsystem tacl (USER attribute ALLOWED-AUTHENTICATIONS lists subsystem tacl) as well as a command interpreter (ALLOW-CI YES). If subsystem is requested by the client (e.g. via ssh -s usr@host tacl), then a TACL process is started after successful authentication and the user sees the TACL prompt. If a shell request is requested by the client (e.g. via ssh usr@host) and the terminal the client was started is of type TN6530 or TN6530-8, then a TACL process is started as well. For any other terminal type a shell request will start a shell under OSS.

The user may request a specific command interpreter by specifying a remote command “tacl -p <program>”, e.g.:

```
ssh usr@host tacl -p fup
```

With a 6530 terminal on the client side the program $SYSTEM.SYSTEM.FUP is started (actual object FUP found on the SYSnn subvolume) and the user sees a FUP prompt and can enter any number of FUP commands. The session ends after the user entered the FUP command EXIT.

It is possible to specify a command for the requested command interpreter via “tacl -p <program> <command>”. For example, when executing the following command,

```
ssh usr@host tacl -p fup info
```

a FUP is started, the FUP command INFO is executed and the session ends.

Even though USER attribute ALLOW-CI-PROGRAM-OVERRIDE is set to NO in the default configuration, the above commands work. The reason is that subsystem tacl is allowed in the default USER configuration, i.e. a user can request subsystem tacl, gets the TACL prompt and can execute the <program> (FUP in the example) anyway. Therefore the value of attribute ALLOW-CI-PROGRAM-OVERRIDE is ignored in this case.

Configuration with Subsystem TACL not Allowed

Since version 0097 it is possible to start a command interpreter even when subsystem tacl is not allowed (USER attribute ALLOWED-AUTHENTICATIONS does not list subsystem tacl). Before version 0097, the execution of CI-PROGRAM or a command interpreter specified as remote command on the SSH client command line was rejected if tacl was not an allowed subsystem. Now, with ALLOW-CI yes and a 6530 terminal on the client side the program configured under CI-PROGRAM, e.g. $SYSTEM.SYSTEM.FUP can be executed by specifying “ci” on the command line, e.g.:

```
ssh usr@host ci
```

The command interpreter will be started and its prompt appears (the FUP prompt in the example) and the user can execute commands processed by the started command interpreter.

Alternatively, a command can be specified on the ssh command line, e.g.

```
ssh usr@host ci –c info
```

After the command interpreter was started, the specified command gets executed and the session is closed. This works only if CI-COMMAND is not set in the USER configuration. Otherwise the CI-COMMAND gets executed and the command on the SSH client command line is ignored.

The user can specify a program, e.g.

```
ssh usr@host ci –p scf
```

but this will be rejected with error “Command interpreter initialization failed” if ALLOW-CI-PROGRAM-OVERRIDE is NO. After changing the value of this attribute to YES, the above command gets executed and the specified command interpreter starts and its prompt is displayed.

The user may try to start a TACL via the ci feature, e.g. like

```
ssh usr@host ci –p tacl
```

This will be rejected because subsystem TACL is not allowed and granting TACL access via command interpreter access would circumvent the configured subsystem restriction.

Having configured TACL as CI-PROGRAM and ALLOW-CI-PROGRAM-OVERRIDE set to NO, a TACL with a specific command can still be executed, even if subsystem TACL is not allowed. Unless CI-COMMAND is configured, a command can be specified on the SSH client side, e.g.

```
ssh usr@host ci -c fileinfo
```

This is allowed as the user does not get a TACL prompt.

The command could be a TACL macro, e.g. a file with the following content:

```
?TACL MACRO
#OUTPUT Macro %0% started with parameters: >%*%<
```

That macro could be started, for example, using the command below:

```
ssh usr@host ci -c $TEMP.TEMP.MYMACRO
```

The TACL process that gets started will display something like the following:

```
…
$TEMP.TEMP.MYMACRO abc def 123
Macro $TEMP.TEMP.MYMACRO started with parameters: >abc def 123<
```

It is also possible to set CI-COMMAND to "$TEMP.TEMP.MYMACRO abc def 123" to avoid the requirement to specify the macro name on the client side. In this case the client command for executing the macro with fixed parameters "abc def 123" would just be as shown below:

```
ssh usr@host ci
```

In cases where a TACL macro should be started but some input from the client side is needed, then it is possible to access the command specified on the client side. If CI-COMMAND is configured, then the specified client side command will not be executed but the command in CI-COMMAND. The command specified on the client side is put into PARAM SSH-ORIGINAL-COMMAND and can be accessed by the TACL macro.

Example content of a macro making use of that PARAM:

```
?TACL MACRO
#OUTPUT Macro %0% started with parameters: >%*%<
#OUTPUT SSH-ORIGINAL-COMMAND was: >[#PARAM SSH-ORIGINAL-COMMAND]<
```

If the command 'test data' is specified as in:

```
ssh usr@host ci -c some data from client
```

then the output would be similar to:

```
…
$TEMP.TEMP.MYMACRO abc def 123
Macro $TEMP.TEMP.MYMACRO started with parameters: >abc def 123<
SSH-ORIGINAL-COMMAND was: >ci -c some data from client<
```

Please remember that through this section the assumption is that a 6530 terminal is on the client side.

The SSH User Database

Overview of SSH Operation Modes

As explained in the Introduction, the SSH2 process accesses a database to …

- discover allowed operations for remote users as well as their logon credentials when running as SSH daemon, allowing remote systems running an SSH or SFTP client to connect to the local NonStop system. This mode of operation is referred to as "daemon mode" within this chapter.
- find local system users' key files and remote host public keys when SSH and SFTP clients on the NonStop system connect to remote systems running an SSH/SFTP implementation. This mode of operation is referred to as "client mode" within this chapter.

This chapter describes the content of the database for both modes and shows how to create and maintain the database. While all database content is kept in a single file, the content of the database is distinctly different for the daemon and client mode:

- In daemon mode, the SSH2 process allows remote SFTP clients to connect to the NonStop system. The database therefore contains remote user credentials as well as public keys of remote systems. See the next section for a detailed description of the database content in daemon mode.
- In client mode, the SSH2 process will connect to remote systems and authenticate NonStop users on the remote system. To do so, the SSH2 process will map NonStop user ID's to private key files stored in the database. It also keeps public keys of known hosts in the database in order to authenticate the remote system. See the section entitled "Database for Client Mode" for details about the database content in client mode.

In order to separate the two different "sections" of the database, the SSHCOM command interpreter, which is used to maintain the database, implements a MODE command that is used to switch between maintaining the data base content for daemon and client modes.

To maintain the daemon database content, issue the following command within SSHCOM:

```
% MODE DAEMON
```

or, because SERVER is supported as alternative for DAEMON:

```
% MODE SERVER
```

To maintain the client database content, issue the following command:

```
% MODE CLIENT
```

Database for Daemon Mode

Format and Content of the Database

In daemon mode, the SSH2 database contains USER and RESTRICTION-PROFILE entities controlling the way incoming ssh connections are processed. The USER records mainly define the allowed authentication methods and the mapping from SSH user to a local Guardian user or alias but also contain other attributes, e.g. for defining access restrictions and use of resources. The following information is held for remote users accessing the NonStop SSH/SFTP service remotely (field names to be used in administration of the database are shown in bold at the beginning of each entry).

The USER entity has the following properties:

- **USER**: The ssh user name used at the remote end of the connection.
- **COMMENT**: Comment text for the ssh user.
- **ALLOWED-AUTHENTICATIONS**: The authentication mechanisms that are allowed for the ssh user.
- **PRINCIPAL**: Kerberos/GSSAPI related attribute: remote principal name configured for ssh user.
- **OWNER**: An existing local system user allowed to modify the USER record. The allowed actions of the owner of a record and the manager of the owner of the record are be the same as defined by PARTIALSSHCOMACCESSUSER/GROUP parameters.
- **SYSTEM-USER**: The local Guardian user name or alias under which operations initiated by the remote user will be executed.
- **PUBLICKEY**: One or more public key(s) sent by the remote user for authentication (see chapter "SSH Protocol Reference" for details). The secret part of the Public Key pair is not configured in USER records. Several attributes are defined for each PUBLICKEY (name, fingerprint, last modified and last used date).
- **ALLOW-SHELL**: Indicating if the ssh user is allowed to request a shell.
- **SHELL-PROGRAM**: OSS path of the shell executed when the ssh user requests a shell or configuration of a telnet service connected to when the ssh user requests a shell.
- **SHELL-COMMAND**: Enforced shell command executed when the ssh user requests a shell.
- **SHELL-ENVIRONMENT**: Pathname of a script that will be executed when a shell is invoked.
- **ALLOW-CI**: Indicating if the ssh user is allowed to request a TACL command interpreter.
- **ALLOW-CI-PROGRAM-OVERRIDE**: Indication if the ssh user is allowed to override the configured CI-PROGRAM via "tacl -p" or "ci -p" command.
- **CI-PROGRAM**: Guardian object name of the command interpreter executed when the ssh user requests a command interpreter or configuration of a telnet service connected to when the ssh user requests a command interpreter.
- **CI-COMMAND**: Startup parameters for CI-PROGRAM used when the ssh user requests a command interpreter.
- **ALLOW-PTY**: Indicating if the ssh user is allowed to request a pseudo terminal (PTY).
- **PTY-SERVER**: User specific configuration of the PTY server process. Ignored if ALLOW-PTY is set to NO. Default value is taken from SSH2 parameter PTYSERVER.
- **ALLOW-TCP-FORWARDING**: Indicating if the ssh user is allowed to request port forwarding.
- **ALLOWED-SUBSYSTEMS**: Subsystems the ssh user is allowed to request.
- **ALLOW-GATEWAY-PORTS**: Indicating if the ssh user is allowed to open gateway ports, i.e. port forwarding where the listen is made on an interface that is not the loopback network interface.

- **ALLOW-MULTIPLE-REMOTE-HOSTS**: Indicating if the ssh user is allowed to connect from multiple remote hosts (a remote host is identified by its IP address).
- **RESTRICTION-PROFILE**: Name of restriction profile defining restrictions regarding incoming connections for the ssh user.
- **PRIORITY**: Priority for a specific ssh user's non-SFTPSERV processes. If omitted, the priority of the SSH2 process is used as default value.
- **CPU-SET**: List of CPUs ssh user's non-SFTPSERV processes are started in.
- **SFTP-INITIAL-DIRECTORY**: The initial directory the remote user will see after successful logon.
- **SFTP-GUARDIAN-FILESET**: List of Guardian filename patterns identifying the files the ssh user can access in a SFTPSERV session.
- **SFTP-SECURITY**: A set of operations the remote user is allowed to perform (i.e. Read, Write, Purge).
- **SFTP-PRIORITY**: This attribute is used to pre-set the priority for a specific user's SFTPSERV processes. If omitted, the default priority of 100 is used.
- **SFTP-CPU-SET**: List of CPUs ssh user's SFTPSERV processes are started in.
- **STATUS**: Status of the USER record.

The USER entity also contains some additional information collected by SSH2 about each ssh user:

- **LAST-LOGON**: Time of last logon.
- **LAST-UNSUCCESSFUL-ATTEMPT**: Time of last failed logon attempt.
- **LAST-AUTH-METHOD**: Authentication method used for last logon.
- **LAST-PUBLICKEY**: Name of last public key (configured in USER record for incoming connections) used in last public key authentication.
- **LAST-IP-ADDRESS**: IP address the last incoming connection was initiated from.
- **LAST-MODIFIED**: Record maintenance: Last time the record was modified.

Each PUBLICKEY entry of a USER entity contains the following attributes:

- **PUBLICKEY NAME**: a free text field allowing you to enter a descriptive comment
- **COMMENT**: a free text field allowing you to enter a descriptive comment
- **MD5**: The MD5 fingerprint of the public key.
- **BABBLE**: The bubble-babble fingerprint of the public key.
- **CREATION-DATE**: the time the key was added to the USER record. A key is in state 'PENDING' if LIVE-DATE has not been reached yet.
- **LIVE-DATE**: the time the key changes or has changed to state 'LIVE'. If the attribute LIVE-DATE is not set, then a key is automatically in state 'LIVE'. A key stays in this state until EXPIRE-DATE is reached.
- **EXPIRE-DATE**: the time the key changes or has changed to state 'EXPIRED'.
- **LIFE-CYCLE-STATE**: the life-cycle state the user public key is in. Possible values are 'PENDING', 'LIVE' and 'EXPIRED'. This is actually not an explicit database field but its value will be determined by the three database fields CREATION-DATE, LIFE-DATE and EXPIRE-DATE.

The database also contains some additional information collected by SSH2 about each public key:

- **LAST-USE**: Key usage: Last time the public key was used.
- **LAST-MODIFIED**: Maintenance: Last time the public key entry was modified.

The RESTRICTION-PROFILE entity has the following properties:

- **RESTRICTION-PROFILE**: The name for the restriction profile, referenced by a USER entity.
- **COMMENT**: Comment text for the restriction profile.
- **CONNECT-FROM**: IP addresses the user is allowed to connect from.
- **CONNECT-TO**: IP addresses a user is allowed to connect to.
- **PERMIT-LISTEN**: Local ports the user is allowed to use for port forwarding.
- **PERMIT-OPEN**: Target host and port combinations the user is allowed to use for port forwarding.
- **FORWARD-FROM**: Remote hosts the user can access ssh tunnels from.
- **LAST-MODIFIED**: Record maintenance: Last time the record was modified.

Database for Client Mode

Format and Content of the Database

In client mode, the SSH2 database contains three entities, which are all related to a local Guardian system user:

- **KEY**s are private user keys used to authenticate to remote systems.
- **PASSWORD**s are passwords used to authenticate to remote systems
- **KNOWNHOST**s are remote systems that are authenticated by configuring their IP addresses, port numbers, and public keys

All three entities contain a set of properties that are used when a local Guardian system user initiates an outgoing connection. Access to the client mode records is controlled by the local Guardian user name, which is stored in client mode records.

Client mode record type KEY holds user key information for the local Guardian user initiating a client connection on NonStop. The key information in the client mode database includes the complete Public Key pair, i.e. both public and private part. KEY records are created via SSHCOM command GENERATE KEY. Database key to the KEY entity consists of:

- **KEY**: the name of the public key pair generated for the Guardian user
- **USER**: the name of the local Guardian user the public key was generated for

The KEY entity has the following additional properties:

- **COMMENT**: a free text field allowing you to enter a descriptive comment
- **TYPE**: The type of the key, supported key types are RSA and DSA
- **BITS**: The number of bits of the key.
- **PUBLICKEY-FINGERPRINT**: The fingerprints of the public key associated with that private key.
- **STATUS**: whether the key is frozen or thawed.
- **CREATION-DATE**: the time the key was generated, if available. A key is in state ‘PENDING’ if LIVE-DATE has not been reached yet.
- **LIVE-DATE**: the time the key changes or has changed to state ‘LIVE’. If the attribute LIVE-DATE is not set, then a key is automatically in state ‘LIVE’. A key stays in this state until EXPIRE-DATE is reached.
- **EXPIRE-DATE**: the time the key changes or has changed to state ‘EXPIRED’.

- **LIFE-CYCLE-STATE**: the life-cycle state the user private key is in. Possible values are ‘PENDING’, ‘LIVE’ and ‘EXPIRED’. This is actually not an explicit database field but its value will be determined by the three database fields CREATION-DATE, LIFE-DATE and EXPIRE-DATE.

The database also contains some additional information collected by SSH2 about each key record:

- **LAST-USE**: Record usage: Last time the record was used.
- **LAST-MODIFIED**: Record maintenance: Last time the record was modified.

Client mode record type PASSWORD holds user password information for the Guardian user initiating a client connection on NonStop. PASSWORD records are added when a user confirms a password is to be stored or via SSHCOM command ADD PASSWORD. Database key to the PASSWORD entity consists of:

- **USERID@HOST**: the user name sent to the remote system and the IP address and port of the remote system.
- **USER**: the name of the Guardian user the public key was generated for

The PASSWORD entity has the following additional properties:

- **STATUS**: whether the password is frozen or thawed.

The database also contains some additional information about each password record collected by SSH2:

- **LAST-USE**: Record usage: Last time the record was used.
- **LAST-MODIFIED**: Record maintenance: Last time the record was modified.

Client mode record type KNOWNHOST holds remote host key information for the Guardian user initiating a client connection on NonStop. KNOWNHOST records are added when a user accepts a remote host key or via SSHCOM command ADD KNOWNHOST. Database key to the KNOWNHOST entity consists of:

The KNOWNHOST entity has the following properties:

- **KEY**: the name of the public key pair generated for the Guardian user
- **KNOWNBY**: the name of the Guardian user who is allowed to connect to this host (or who accepted the remote host key when SSH2 parameter STRICTHOSTKEYCHECKING is set to FALSE). The special name 'all' is supported indicating that the remote host key is configured for all users.

The KNOWNHOST entity has the following additional properties:

- **COMMENT**: a free text field allowing you to enter a descriptive comment.
- **ADDRESSES**: the IP addresses or DNS names of the hosts using this public key.
- **PORT**: the port number of the SSH daemons running on the remote host.
- **ALGORITHM**: the algorithm used for host authentication. Valid algorithms are SSH-RSA and SSH-DSS.
- **PUBLICKEY-FINGERPRINT**: The MD5 and bubble-babble fingerprints of the public key.
- **STATUS**: whether the knownhost is frozen or thawed.

The database also contains some additional information collected by SSH2 about each knownhost:

- **LAST-USE**: Record usage: Last time the record was used.
- **LAST-MODIFIED**: Record maintenance: Last time the record was modified.

Creating and Accessing the Database

The database is contained in a single Enscribe file. To create a new database, SSH2 needs to be started with the SSHCTL parameter pointing to a non-existing file. In that case, the SSHCTLAUDIT parameter will control whether the database will be created as an audited file or not.

To reuse an existing database, SSH2 needs to be started with SSH2 parameter SSHCTL pointing to an existing file.

The content of the database is viewed and maintained with the SSHCOM utility, which is described in the next section.

Exporting the Database

The SSHCTL database can be exported into text files in order to allow further processing of the content. The text files are written in standard comma-separated form, which allows importing of the text files into speadsheet and database programs or any SQL database.

For a description how to export the database please refer to the section "Miscellaneous commands in SSHCOM" in chapter "SSHCOM Reference".

Copying the Database

After copying the SSH database file you may need to alter table records depending on the requirements of the new SSH environment.

The commands to alter attributes of existing records or to delete or add records are discussed in the next section.

SSHCOM Command Reference

SSHCOM Overview

SSHCOM is a command interpreter delivered with the SSH2 component. It is used to view and maintain the SSH2 user database. Using SSHCOM is similar to working with the HP PATHCOM utility. You connect to an existing SSH2 process using the OPEN command, then you issue commands against that instance of SSH2, which will access the corresponding area in the database. Please see section "Overview of SSH Operation Modes" for an explanation for the logical separation of those database entities that are related to outgoing connections (client mode entities) and database entities that are related to incoming connections.

SSHCOM commands can be continued over multiple lines. When an ampersand ("&") appears as the last character on a line, the command is continued with the first column of the next line. There is no limit on the number of lines over which a command may be continued, but commands are limited to 10240 characters. Prior to STN version B24 the limit was 1024 characters. Note that SSHCOM and STNCOM have the same code base. If SSHCOM is prompting at a terminal for input, the prompt for continuation lines will be the current prompt prefixed by ampersand ampersand space: "&& ". Continuations are allowed from terminals, IN files and OBEY files.

SSHCOM is started with a simple TACL command. After switching to the proper mode (see "Overview of SSH Operation Modes" in the chapter "The SSH User Database"), the HELP command will give you a brief overview of the supported commands. Note that the HELP command will result in a different output in the two modes. The following example shows the output in client mode:

```
$QAHPSSH T0801ABK 3> run sshcom $ssh01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:18:49.958
OPEN $ssh01
% mode client
mode client
OK, switched to client mode
% help
help
+-----------------------Main Menu-------SSHCOM CLIENT Mode--------------------+
|                                                                             |
| Client Mode Commands:                                                       |
| =====================                                                       |
|                                                                             |
| Commands operating on KEY entity:                                           |
| ---------------------------------                                           |
| ALTER              DELETE             EXPORT             FREEZE             |
| GENERATE           IMPORT             INFO               RENAME             |
| THAW                                                                        |
|                                                                             |
| Commands operating on KNOWNHOST entity:                                     |
| ---------------------------------------                                     |
| ADD                ALTER              DELETE             FREEZE             |
| INFO               RENAME             THAW                                  |
|                                                                             |
| Commands operating on PASSWORD entity:                                      |
| ---------------------------------------                                     |
| ADD                ALTER              DELETE             FREEZE             |
```

```
| INFO              RENAME            THAW                                    |
|                                                                           |
| General Commands                                                          |
| ----------------------                                                    |
| INFO SSH2         INFO SYSTEM-USER                                        |
|                                                                           |
| Miscellaneous Commands                                                    |
| ----------------------                                                    |
| ASSUME            EXIT              MODE              OBEY                |
| PAUSE             PROMPT            TIME                                  |
|                                                                           |
+------------------------------SecurFTP/SSH Modes-----------------------------+
|                                                                           |
| CLIENT            DAEMON                                                  |
|                                                                           |
+---------------------------------------------------------------------------+
%
```

Use command HELP MODE to find out more about modes.

The following example shows the output in daemon mode:

```
% mode daemon
mode daemon
OK, switched to daemon mode
% help
help
+----------------------Main Menu-------SSHCOM DAEMON Mode--------------------+
|                                                                           |
| Daemon Mode Commands:                                                     |
| =====================                                                     |
|                                                                           |
| Commands operating on USER entity:                                        |
| ----------------------------------                                        |
| ADD               ALTER             DELETE            FREEZE              |
| INFO              RENAME            THAW                                  |
|                                                                           |
| Commands operating on RESTRICTION-PROFILE entity:                         |
| -------------------------------------------------                         |
| ADD               ALTER             DELETE            INFO                |
| RENAME                                                                    |
|                                                                           |
| General Commands                                                          |
| ----------------------                                                    |
| ABORT             CLEAR             ENABLE            DISABLE             |
| FLUSH             INFO DEFINE       INFO SSH2         RESET               |
| RESOLVE           ROLLOVER          SET               STATISTICS          |
| STATUS                                                                    |
|                                                                           |
| Miscellaneous Commands                                                    |
| ----------------------                                                    |
| EXIT              EXPORT            MODE              OBEY                |
| PAUSE             PROMPT            TIME                                  |
|                                                                           |
+------------------------------SSH2 Modes------------------------------------+
|                                                                           |
| CLIENT            DAEMON     <-- Use HELP MODE to find out about modes    |
|                                                                           |
+---------------------------------------------------------------------------+
%
```

Standard NonStop™ Commands and Features

The following NonStop™ Guardian standard commands and features are supported in SSHCOM:

- FC command to modify the last command used.
- OBEY command to obey a set of commands contained in an EDIT file.

- Processing of a file through the standard TACL way of RUN SSHCOM /IN file/.
- Pausing the display with the PAUSE command.
- Line continuation through the usage of the "&" character.

Standard behavior is that for each command entered a message is displayed about the outcome, i.e. if the command succeeded or failed (if no message is displayed it should be assumed that the command could not be parsed successfully).

It is possible to add comments in IN files, OBEY files and at the interactive prompt. Any text following an exclamation mark is considered as comment text. A comment line is continued on the next line if the last character is an ampersand.

Note: A single exclamation mark alone entered at the SSHCOM terminal prompt means "repeat last command unchanged" while a single exclamation mark in an IN or OBEY file is treated as comment line.

Startup Values for the MODE and ASSUME USER Commands

When being started from TACL, SSHCOM applies some heuristics to set the startup values for the MODE and ASSUME USER commands. (The ASSUME USER command is described later in subsection "Client Mode Commands - Introduction"). It will determine the startup values as follows:

- If SSHCOM is started by the Guardian User SUPER.SUPER, it will set DAEMON mode and assume the user SUPER.SUPER.
- For any other user, CLIENT mode will be set and that user will be assumed.

Security within SSHCOM

SSHCOM implements security by checking the user who has started SSHCOM from TACL.

The following commands are considered sensitive and can only be executed from users or groups who are explicitly given full SSHCOM access:

- Exporting any private key with the EXPORT KEY,..,PRIVATE command. This means that the private key of the user, for instance COMF.MH, can only be exported by users with full SSHCOM access — not even by the user COMF.MH (unless user COMF.MH was given full SSHCOM access).
- Commands operating on client mode entities that are associated with a user other than the user starting SSHCOM.
- Commands operating on daemon mode entities.

Configuration of Users with Full SSHCOM Access

There are two ways for allowing full SSHCOM access:

- Create a Safeguard OBJECTTYPE USER record or
- Set parameter sets FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j>

The existence of an OBJECTTYPE USER record overwrites any FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j> configuration.

Only super.super user has full access to all SSHCOM commands if there is no thawed OBJECTTYPE USER record defined and none of the above mentioned parameter sets are defined.

User super.super does not have full SSHCOM access only if explicitly denied Create authority in a thawed OBJECTTYPE USER record.

The following sections explain the SSHCOM access rights in more detail.

Dependency on Safeguard OBJECTTYPE USER Record

Every administrator that configures an OBJECTTYPE USER record is highly aware of the importance and relevance of USER configuration on NonStop systems. But some may not be fully aware that the SSH configuration is a highly critical, security-relevant task as well: A user that is allowed to configure SSH USER records can create access to the NonStop system without Safeguard authentication, i.e. configuring SSH USER records is as critical as configuring Safeguard USER records.

If a user is denied executing Safeguard SAFECOM ADD/ALTER USER commands, then this user must be denied ADD/ALTER USER in SSHCOM in order to ensure a consistent security policy.

Starting with release 89 there is tighter coupling of SSHCOM security with Safeguard security. This does not only include checking if a Safeguard user is frozen (see section "ALLOWFROZENSYSTEMUSER") but also includes support of OBJECTTYPE USER (please refer to HP NonStop™ manuals "Safeguard Reference Manual" and "Safeguard Administrator's Manual").

The current implementation ignores OBJECTTYPE USER ACL entries containing a network id (\node-spec). The SSH2 process issues a warning message if it finds such an entry. Another restriction is that only the primary group of a user is checked against group based OBJECTTYPE USER ACL entries.

In order to reduce overhead the OBJECTTYPE USER, USER and ALIAS information retrieved from SafeGuard is cached. It can take up to 5 minutes before an SSH2 process takes SafeGuard modifications into account. By restarting an SSH2 process any SafeGuard changes will be active in the SSH2 process immediately.

SSHCOM Security without Safeguard OBJECTTYPE USER Record

If a Safeguard OBJECTTYPE USER record does not exist or exists but is frozen, the behavior is as follows:

DAEMON MODE commands

The user super.super can execute any daemon mode commands. The parameter sets FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j> are evaluated and users and groups configured in these parameter sets are granted full access to all daemon mode commands.

CLIENT MODE commands

The user super.super can execute any client mode command for any user. The parameter sets FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j> are evaluated and configured users and groups are granted full access to all client mode commands for any user.

If a person that is not logged on as super.super and not configured in parameter sets FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j> wants to execute an SSHCOM CLIENT MODE command affecting records for a specific Guardian user or alias <user-or-alias> must either be logged on as <user-or-alias> or meet these two qualifications:

- Be the group manager of the underlying Safeguard user ID
- Be the owner of the underlying Safeguard user ID of <user-or-alias> or be the group manager of the owner of the underlying Safeguard user ID of <user-or-alias>

SSHCOM Security with existing Safeguard OBJECTTYPE USER Record

If a Safeguard OBJECTTYPE USER record exists and is not frozen, the behavior is as follows:

DAEMON MODE commands

The user super.super can execute any daemon mode commands unless explicitly configured in the OBJECTTYPE USER with DENY Create authority. The parameter sets FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j> are ignored. Non-super.super users configured with Create authority in the OBJECTTYPE USER record are granted full access to all daemon mode commands.

CLIENT MODE commands

The user super.super can execute any client mode commands for all users unless explicitly configured in the OBJECTTYPE USER with DENY Create authority. The parameter sets FULLSSHCOMACCESSUSER<i> and FULLSSHCOMACCESSGROUP<j> are ignored.

If a person wants to execute an SSHCOM CLIENT MODE command affecting records for a specific Guardian user or alias <user-or-alias> must either be logged on as <user-or-alias> or meet these two qualifications:

- Have CREATE (C) authority on the OBJECTTYPE USER access control list
- Be the owner of the underlying Safeguard user ID of <user-or-alias> or be the group manager of the owner of the underlying Safeguard user ID of <user-or-alias>

SSHCOM Access Summary

Shortcuts used in the following table:

- **'SUPER'** - SUPER.SUPER
- **'OU'** - OBJECTTYPE USER
- **'OUR'** - OBJECTTYPE USER RECORD
- **'FullSA'** - FULLSSHCOMACCESSUSERi/GROUPj
- **'PartialSA'** - PARTIALSSHCOMACCESSUSERk/GROUPn

User is 'SUPER' (Yes/No)	**Thawed 'OU' exists (Yes/No)**	**User configured in 'OUR' (No / Create / DENY Create / Not Applicable)**	**User included in 'FullSA' configuration (Yes / No / Not Applicable)**	**User included in 'PartialSA' configuration (Yes/No)**	**Allowed USER Commands (All / Alter&Info / None)**
Yes	No	N/A	N/A	N/A	All
Yes	Yes	No	N/A	N/A	All
Yes	Yes	Create	N/A	N/A	All
Yes	Yes	DENY Create	N/A	No	None
Yes	Yes	DENY Create	N/A	Yes	Alter&Info
No	No	N/A	No	No	None
No	No	N/A	No	Yes	Alter&Info
No	No	N/A	Yes	N/A	All
No	Yes	No	N/A	No	None
No	Yes	No	N/A	Yes	Alter&Info
No	Yes	Create	N/A	N/A	All
No	Yes	DENY Create	N/A	No	None
No	Yes	DENY Create	N/A	Yes	Alter&Info

Ownership and Management of Client Mode Entities

In release 89 a finer granularity for access and administration of mode client records was introduced. In previous releases client mode records were owned by a Guardian user identifier. Even when logged on as alias the underlying Guardian identifier was used to add and retrieve KEY, PASSWORD and KNOWNHOST records. The philosophy behind this assumed that one person used a specific Guardian user identifier as well as the configured aliases for that Guardian user identifier. This approach is consistent with the general security on NonStop (ACL, file security, etc.), which is based on the Guardian user identifier.

As each alias has its own password it is possible to create a NonStop environment where different persons use different aliases pointing to the same Guardian user identifier. In such an environment storing KEY, PASSWORD and KNOWNHOST records under the same user id represents a security problem:

Assuming aliases a1 and a2 exist, both configured with underlying Guardian user identifier grp1.usr1. If alias a1 stored a password for remote host h1 and remote user u1 in the client mode database (under grp1.usr1), then alias a2 can connect to host h1 specifying remote user u1 using the stored password entry, i.e. alias a2 gets access to remote host h1 without knowing the password of remote user u1.

In order to resolve this problem a new parameter CLIENTMODEOWNERPOLICY was introduced in release 89 defining the policy how to set the owner of an entry. Defined values are LOGINNAME, GUARDIANNAME and BOTH. The differences are explained in the following sections.

Guardian Users in the Context of SSH Access Policy Explained

In the SSH access policy context we used a variety of terms for users and access. The following text will explain the definitions of these terms and its origin.

An example of a TACL STATUS DETAIL command shows for a process:

```
Userid: 255,255    (SUPER.SUPER)

Login name: root-ssh
```

Every process consists of a "Userid" and "Login name".

The value of "Userid" refers to Guardian user identifier or just guardian user id. The "Userid" is used to do SSH policy access checks when the parameter option GUARDIANNAME is used. In the example above this is 255.255

The value of "Login name" can be a Guardian user id or an alias. The "Login name" is used to do SSH policy access checks when the parameter option LOGINNAME is used. In the example above an alias of root-ssh was used.

In Safeguard an alias is just an alternate name for a user. But the customers sometimes use different alias names that are all assigned to the same underlying Guardian user ID. This presented a huge security hole if an alias was not used as an alternate name (i.e. a human owns both alias and underlying Guardian user) but as a unique user name with a different human being behind each alias.

Please refer to the Safeguard reference manual on the features of the Safeguard security-management.

Client Mode Owner Policy LOGINNAME

The default owner is the login name, which can be a Guardian user identifier or an alias. An alias user cannot add/read/manipulate entries for the Guardian user the alias is configured with; vice versa, a Guardian user also can not add/read/manipulate entries for associated aliases. In other words, a Guardian or alias user can add/manipulate entries for that Guardian or alias user only.

The value LOGINNAME is recommended if different people are using the various aliases configured with the same Guardian user identifier.

Client Mode Owner Policy GUARDIANNAME

The default owner is the Guardian user identifier, independent if the logon name is an alias or a Guardian user. Entries are read using the Guardian user ID only. This means that a Guardian user can add/read/manipulate entries for associated alias users, and vice versa.

The assumption is that the same person uses the aliases of a Guardian user identifier and the Guardian user identifier itself. This was the default before this enhancement was introduced (in release 89) and therefore value GUARDIANNAME needs to be used if the client mode policy of previous releases should be kept.

Client Mode Owner Policy BOTH

The default owner is the login name but a guardian user can add or manipulate entries stored under an alias or a guardian user identifier. Entries are read for both the login name and the guardian user in case these are different (entries of the alias are read first, then entries of the guardian id). The value BOTH is only recommended if a guardian user and all aliases configured for this guardian user are solely used by one person and client mode records are to be stored under Guardian user identifier as well as alias names.

Example: Assume, an alias entry is present, but not an entry for the associated Guardian ID, and the user is logged on as the alias. With client mode owner policy set to LOGINNAME, privileges to read/alter the entry would be granted, for GUARDIANNAME they would not be granted because a matching entry is not found, and for BOTH they would be granted. If the Guardian entry is present but not the alias, and the user is logged on as the alias, LOGINNAME access would not be allowed, GUARDIANNAME would be allowed, and BOTH would also be allowed.

Client Mode Owner Policy Examples

Assuming Guardian User SUPER.MARIO and alias super-m are configured in Safeguard:

```
=info alias super-m

NAME                                USER-ID     OWNER                          STATUS

super-m                             255,20     254,255                         THAWED

=info user super.mario

GROUP.USER          USER-ID    OWNER     LAST-MODIFIED     LAST-LOGON      STATUS

SUPER.MARIO         255,20     254,255   12FEB11, 22:36 16FEB13, 13:50   THAWED
```

An alias entry is present in the SSH database, but not an entry for the associated Guardian ID, e.g.:

```
% info key *:*

info key *:*

KEY                        TYPE USER                 LIFE-CYCLE LAST-USE       STATUS
k1                          RSA super-m              LIVE       *NONE*         THAWED
```

Assuming the user is logged on as the alias super-m. With client mode owner policy set to LOGINNAME, privileges to read/alter the entry k1 would be granted, for GUARDIANNAME they would not be granted because a matching entry is not found, and for BOTH they would be granted.

If the Guardian entry is present but no entry for the alias, e.g.:

```
% info key *:*
info key *:*

KEY                        TYPE USER                 LIFE-CYCLE LAST-USE       STATUS
k2                          RSA SUPER.MARIO          LIVE       *NONE*         THAWED
```

and the user is logged on as the alias super-m, then access to entry k2 would not be denied with client mode owner policy set to LOGINNAME but would be allowed with client mode owner policy set to GUARDIANNAME or BOTH.

Note: The default value for CLIENTMODEOWNERPOLICY is BOTH. Please be aware that the default client mode policy changed from GUARDIANNAME to BOTH with release 89. This change of the policy should not cause problems with existing records as records had been read in previous releases only if stored under the Guardian user identifier (entries stored under an alias had been ignored).

The following will change when using the new default value BOTH or value LOGINNAME:

If a user is logged on as an alias and new CLIENT MODE records are added (PASSWORD, KNOWNHOST, PUBLICKEY), then the new records will be stored under the alias name. An alias user is not allowed to add records for the underlying Guardian user when CLIENTMODEOWNERPOLICY is set to LOGINNAME.

Client Mode Owner Policy and Processing of SSHCOM Commands

The processing of the CLIENT mode SSHCOM commands has been enhanced in release 89 to support the new CLIENTMODEOWNERPOLICY values LOGINNAME and BOTH. If the value is set to either LOGINNAME or BOTH the following applies:

- Entries can be added with alias user names. A user logged on using an alias can only display, add, and manipulate entries for that alias.
- A guardian user can display, add, and manipulate entries for the Guardian user.
- Depending on the rules explained in the section about OBJECTTYPE USER records a group manager can add, change or delete client mode records stored under an alias or Guardian name.
- A user with full access can add/manipulate all entries unless an OBJECTTYPE USER record says otherwise.

If parameter CLIENTMODEOWNERPOLICY is set to value GUARDIANNAME, then the following applies:

- Any attempt to add entries under an alias name will be rejected. Entries will be added under the Guardian name.
- A guardian user can display, add, and manipulate entries for the Guardian user.
- Depending on the rules explained in the section about OBJECTTYPE USER records a group manager can add, change or delete client mode records stored under a Guardian name.
- A user with full access can add/manipulate all entries unless an OBJECTTYPE USER record says otherwise.

Miscellaneous commands in SSHCOM

The following commands are independent of the mode set with the mode command:

MODE

As described earlier, the MODE command will work in both run modes of SSHCOM. If entered without specifying a mode, the command will show the current mode under which SSHCOM is operating:

```
$QAHPSSH T0801ABK 29> run sshcom $ssh01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% mode
Mode
current mode is CLIENT
%
```

The command has the following syntax:

```
MODE [CLIENT | DAEMON | SERVER]
```

The individual attributes have the following meaning and syntax:

`CLIENT`

Switches to CLIENT mode.

`DAEMON`

Switches to DAEMON mode.

`SERVER`

SERVER is a synonym for DAEMON and therefore switches to DAEMON mode as well.

SET

The SET command allows you to change some configuration parameters during runtime. Currently the following parameters are supported:

Parameter	Meaning
AUDITCONSOLE	Determines whether audit messages are written to the console.
AUDITEMS	Determines whether audit messages are written to EMS.
AUDITFILE	Determines whether audit messages are written to a file.
AUDITFORMATCONSOLE	Controls the format of the audit messages that are written to the console.
AUDITFORMATEMS	Controls the format of the audit messages that are written to EMS.
AUDITFORMATFILE	Controls the format of the audit messages that are written to a file.
LOGCACHEDUMPONABORT	Determines if the internal log cache is written to the log file in case of process aborting.
LOGCACHESIZE	Determines the size of the internal log cache.
LOGCONSOLE	Determines whether log messages are written to a console.
LOGEMS	Determines whether log messages are written to EMS.
LOGFILE	Determines whether log messages are written to a file.
LOGFORMATCONSOLE	Controls the format of the log messages that are written to the console.
LOGFORMATFILE	Controls the format of the log messages that are written to a file.
LOGFORMATEMS	Controls the format of the log messages that are written to EMS.
LOGLEVELCACHE	Determines whether log messages are written to the internal log cache.
LOGLEVELCONSOLE	Determines which messages will be written to the console.
LOGLEVELFILE	Determines which messages will be written to the log file.
LOGLEVELEMS	Determines which messages will be written to EMS.

Please see the chapter "Monitoring and Auditing", section "Destinations for Log Messages" for a description of those parameters. The following screenshot shows how the LOGLEVELFILE is changed to 70 using the SET command:

```
$QAHPSSH T0801ABK 29> run sshcom $ssh01
SSHCOM T0801H01_22JAN2014_ABK - 2014-01-24 14:42:45.368
OPEN $ssh01
% set loglevelfile 70
set loglevelfile 70
OK, LOGLEVELFILE set to 70
%
```

INFO SSH2

The INFO SSH2 command will display the startup configuration as well as the current settings of all parameters that can be changed using the SET command. The following screenshot shows the output of the INFO SSH2 command, after changing the LOGLEVELFILE with the command shown above (example):

```
% info ssh2
info ssh2
-----------------------------------------------------------
SSH2 version T9999H06_22Jan2014_comForte_SSH2_0097
-----------------------------------------------------------
Startup configuration:
[file ] *                    <log configuration>
[def  ] ALLOWEDAUTHENTICATIONS <keyboard-interactive,password,publickey>
[file ] ALLOWEDSUBSYSTEMS    <sftp,tacl>
[def  ] ALLOWFROZENSYSTEMUSER <FALSE>
[def  ] ALLOWINFOSSH2        <ALL>
```

```
[def  ] ALLOWPASSWORDSTORE   <TRUE>
[file ] ALLOWTCPFORWARDING   <TRUE>
[def  ] AUDITCONSOLE         <*>
[def  ] AUDITEMS             <*>
[file ] AUDITFILE            <$QAHPSSH.T0801ABK.ZTC1AUD>
[file ] AUDITFILERETENTION   <10>
[def  ] AUDITFORMAT          <21>
[def  ] AUDITFORMATCONSOLE   <0>
[def  ] AUDITFORMATEMS       <0>
[def  ] AUDITFORMATFILE      <21>
[file ] AUDITMAXFILELENGTH   <1000>
[def  ] AUTOADDAUTHPRINCIPAL <FALSE>
[file ] AUTOADDSYSTEMUSERS   <TRUE>
[def  ] BACKUPCPU            <NONE>
[def  ] BANNER               <*>
[def  ] BURSTSUPPRESSION     <FALSE>
[def  ] BURSTSUPPRESSIONEXPIRATIONTIME <300>
[def  ] BURSTSUPPRESSIONMAXLOGLEVEL <40>
[def  ] CACHEBURSTSUPPRESSION <FALSE>
[def  ] CIPCOMPATERROR       <*>
[def  ] CIPHERS              <aes256-cbc,twofish256-cbc,twofish-cbc,aes128-
cbc,twofish128-cbc,blowfish-cbc,3des-cbc,arcfour,cast128-cbc>
[def  ] CLIENTALLOWEDAUTHENTICATIONS <none,gssapi-with-
mic,publickey,password,keyboard-interactive>
[file ] CLIENTMODEOWNERPOLICY <GUARDIANNAME>
[def  ] COMPRESSION          <TRUE>
[run  ] CONFIG               <$QAHPSSH.T0801ABK.ztc1cfg>
[def  ] CONFIG2              <*>
[def  ] CONSOLEBURSTSUPPRESSION <FALSE>
[def  ] CPUSET               <>
[def  ] CUSTOMER             <>
[file ] DAEMONMODEOWNERPOLICY <LOGINNAME>
[def  ] DNSMODE              <FIRST>
[def  ] EMSBURSTSUPPRESSION  <FALSE>
[def  ] ENABLESTATISTICSATSTARTUP <FALSE>
[def  ] FILEBURSTSUPPRESSION <FALSE>
[def  ] FULLSSHCOMACCESSGROUP1 <>
[def  ] FULLSSHCOMACCESSUSER1 <>
[def  ] GSSAUTH              <*>
[def  ] GSSGEXKEX            <FALSE>
[def  ] GSSKEX               <TRUE>
[def  ] GUARDIANATTRIBUTESEPARATOR <,>
[def  ] HOSTKEY              <HOSTKEY>
[def  ] HOSTKEYBITS          <1024>
[def  ] HOSTKEYTYPE          <DSA>
[def  ] INTERFACE            <0.0.0.0>
[expl ] INTERFACEOUT         <0.0.0.0>
[def  ] INTERVALLIVEPRIVATEUSERKEY <730>
[def  ] INTERVALLIVEPUBLICUSERKEY <730>
[def  ] INTERVALPENDINGPRIVATEUSERKEY <0>
[def  ] INTERVALPENDINGPUBLICUSERKEY <0>
[def  ] IPMODE               <IPV4>
[def  ] LICENSE              <\BWNS02.$QAHPSSH.T0801ABK.LICENSE>
[def  ] LIFECYCLEPOLICYPRIVATEUSERKEY <DISABLED>
[def  ] LIFECYCLEPOLICYPUBLICUSERKEY <DISABLED>
[def  ] LOGCACHEDUMPONABORT  <TRUE>
[def  ] LOGCACHESIZE         <1024>
[file ] LOGCONSOLE           <*>
[file ] LOGEMS               <*>
[def  ] LOGEMSKEEPCOLLECTOROPENED <TRUE>
[file ] LOGFILE              <$QAHPSSH.T0801ABK.ZTC1LOG>
[file ] LOGFILERETENTION     <10>
[def  ] LOGFORMATCONSOLE     <93>
[def  ] LOGFORMATEMS         <16>
[def  ] LOGFORMATFILE        <93>
[file ] LOGLEVEL             <50>
[def  ] LOGLEVELCACHE        <50>
[def  ] LOGLEVELCONSOLE      <50>
[def  ] LOGLEVELEMS          <20>
[def  ] LOGLEVELFILE         <50>
[file ] LOGMAXFILELENGTH     <1000>
```

```
[def  ] MACS                <hmac-sha1,hmac-md5,hmac-sha1-96,hmac-md5-96>
[file ] OWNER               <RoGeR>
[def  ] PARTIALSSHCOMACCESSGROUP1 <>
[def  ] PARTIALSSHCOMACCESSUSER1 <>
[def  ] PAUTHSUPPRESSIPADDRESS <FALSE>
[run  ] PORT                <12229>
[def  ] PTCPIPFILTERKEY     <*>
[def  ] PTCPIPFILTERTCPPORTS <*>
[file ] PTYSERVER           <$ZPTYK>
[file ] RECORDDELIMITER     <ANY>
[def  ] RESTRICTIONCHECKFAILEDDEFAULT <FALSE>
[file ] SFTPALLOWGUARDIANCD  <FALSE>
[def  ] SFTPCPUSET          <>
[def  ] SFTPEDITLINEMODE    <none>
[def  ] SFTPEDITLINENUMBERDECIMALINCR <1000>
[def  ] SFTPEDITLINESTARTDECIMALINCR <-1>
[file ] SFTPENHANCEDERRORREPORTING <2>
[def  ] SFTPEXCLUSIONMODEREAD <SHARED>
[def  ] SFTPIDLETIMEOUT     <-1>
[def  ] SFTPMAXEXTENTS      <900>
[def  ] SFTPPRIMARYEXTENTSIZE <2>
[def  ] SFTPREALPATHFILEATTRIBUTEECHOED <FALSE>
[def  ] SFTPSECONDARYEXTENTSIZE <100>
[def  ] SFTPUPSHIFTGUARDIANFILENAMES <FALSE>
[def  ] SHELLENVIRONMENT    <>
[def  ] SOCKETKEEPALIVE     <1>
[def  ] SOCKETRCVBUF        <0>
[def  ] SOCKETSNDBUF        <0>
[def  ] SOCKTCPMAXRXMT      <0>
[def  ] SOCKTCPMINRXMT      <0>
[def  ] SOCKTCPRXMTCNT      <0>
[def  ] SOCKTCPTOTRXMTVAL   <0>
[def  ] SSH2PROCESSNAME     <$SSH01>
[def  ] SSHAUTOKEXBYTES     <1073741824>
[def  ] SSHAUTOKEXTIME      <3600>
[file ] SSHCTL              <SSHDBK>
[file ] SSHCTLAUDIT         <FALSE>
[def  ] SSHKEEPALIVETIME    <60>
[def  ] STOREDPASSWORDSONLY  <FALSE>
[file ] STRICTHOSTKEYCHECKING <FALSE>
[run  ] SUBNET              <$ZTC1>
[def  ] SUPPRESSCOMMENTINSSHVERSION <FALSE>
[def  ] TCPIPHOSTFILE       <*>
[def  ] TCPIPNODEFILE       <*>
[def  ] TCPIPRESOLVERNAME   <*>
[def  ] USEDISKFILEPREFIXFORFILENAME <FALSE>
[expl ] USERDATABASEUNDERTMFCONTROL <FALSE>

-----------------------------------------------------------
Current configuration:
LOGCONSOLE          *
LOGEMS              *
LOGFILE             $QAHPSSH.T0801ABK.ZTC1LOG

LOGFORMATCONSOLE    93
LOGFORMATEMS        16
LOGFORMATFILE       93

LOGLEVELCONSOLE     50
LOGLEVELEMS         20
LOGLEVELFILE        50

LOGMAXFILELENGTH    1000
LOGFILERETENTION    10

LOGCACHESIZE        1024 (current number of messages in cache: 0)
LOGLEVELCACHE       0
LOGCACHEDUMPONABORT 1

AUDITCONSOLE        *
AUDITEMS            *
```

```
AUDITFILE            $QAHPSSH.T0801ABK.ZTC1AUD
AUDITFORMATCONSOLE   0
AUDITFORMATEMS       0
AUDITFORMATFILE      21
AUDITMAXFILELENGTH   1000
AUDITFILERETENTION   10

%
```

CLEAR LOGCACHE

If a log cache is written (see parameters LOGLEVELCACHE, LOGCACHESIZE), the command CLEAR LOGCACHE can be used to clear the cache. It has the following syntax:

```
CLEAR LOGCACHE
```

The original content of the log cache is lost when executing this command.

FLUSH LOGCACHE

If a log cache is written (see parameters LOGLEVELCACHE, LOGCACHESIZE), the command FLUSH LOGCACHE can be used to write the content of the log cache to the configured log file (parameter LOGFILE must not be set to a value of * to be able to flush the log cache). It has the following syntax:

```
FLUSH LOGCACHE
```

The log cache will be automatically cleared after the content of the log cache was written to the current log file.

INFO DEFINE

The INFO DEFINE command displays information about the DEFINEs as they exist in the SSH2 process context. It has the following syntax:

```
INFO DEFINE { ALL | <define-name> }
```

Especially the TCP/IP defines are relevant because the SSH2 process directly communicates with a TCP/IP process and not the SSH[OSS]/SFTP[OSS] clients themselves.

When ALL is specified, all defines in the SSH2 process context are displayed; otherwise the information is displayed for the specified <define-name>.

OUT <filename> | STOP

STOP

Output to home teminal

<filename>

If a disc file that does not exist, it is created as file code 101 unstructured and is written as an edit-101 file.

If an existing unstructured disc file with code 101, it is erased and written as an edit-101 file.

If an existing disc file that is not unstructured or not code 101, or a non disc file, then the file is opened and sent lines of output.

PROMPT "<text>"

This command redefines the prompt sent to the terminal for new command input.

<text> may contain any displayable character except quote ("), and may be 1 to 64 characters long. Certain embedded commands (case independent) in <text> are replaced as follows:

- $P – the target process name
- $X – the target expand node name
- $T – target system LCT time in format HH:MM
- $D – target system LCT date in format yyyy/mm/dd

Example:

```
PROMPT "$X.$P $D $T STN> "
\DEV.$STN2 2010/08/06 23:59 STN>
PROMPT "$T $P> "
23:59 $STN2>
```

The default setting is PROMPT "% "

The PROMPT command remains in effect until SSHCOM terminates.

RESOLVE HOST-NAME

This command can be used to test the TCP/IP host name resolving. It has the following syntax:

```
RESOLVE HOST-NAME <host-name>
```

The value for <host-name> must be a name known to a DNS server or configured in a HOSTS file. Output will look like:

```
OK, host name 'hostv4' resolved to 10.20.0.210
```

or, for IPv6 address:

```
OK, host name 'hostv6' resolved to fe80::250:56ff:fea7:4bdc (formatted last 4 bytes as dotted
quad: fe80::250:56ff:254.167.75.220)
```

The TCP/IP defines in the context of the SSH2 process are relevant for host name resolving, not those in the context of SSH client processes. Please see SSHCOM command INFO DEFINE.

ROLLOVER AUDITFILE

This command can be used to force a rollover of the configured audit file. The current audit file will be renamed to an audit archive file and a new audit file is opened if the AUDITFILE parameter is not set to * and the parameter AUDITFILERETENTION is set to a non-zero value. The command has the following syntax:

```
ROLLOVER AUDITFILE
```

The ROLLOVER command can only be executed by super.super (unless explicitly denied in OBJECTTYPE USER record) or a user granted full SSHCOM access.

ROLLOVER LOGFILE

This command can be used to force a rollover of the configured log file. The current log file will be renamed to an archive file and a new log file is opened if the LOGFILE parameter is not set to * and the LOGFILERETENTION parameter is set to a non-zero value. The command has the following syntax:

```
ROLLOVER LOGFILE
```

The ROLLOVER command can only be executed by super.super (unless explicitly denied in OBJECTTYPE USER record) or a user granted full SSHCOM access.

EXPORT SSHCTL

The EXPORT SSHCTL command will export the content of the SSH User Database into as many as six text files. All attributes of the various objects are written in the CSV (comma-separated value) format.

The command has the following syntax:

```
EXPORT SSHCTL, SUBVOL <subvolume> [, WIDTH <width>]
```

The individual attributes have the following meaning and syntax:

```
SUBVOL <subvolume>
```

The files are stored in a subvolume specified by the SUBVOL attribute.

Starting with SPR T0801^ABE, an OSS directory may be specified. If a Guardian subvolume is specified, then Guardian edit files are created and long lines will be wrapped. Files exported to a directory will not be wrapped unless option WIDTH is specified. Specifying OSS paths referring to a Guardian namespace like /G/system/ssh2exp leads to code 180 files and no wrapping occurs (if WIDTH is not specified). The volume must be a physical disk in this case.

```
WIDTH <width>
```

Defines the maximum number of characters per output line. If WIDTH is specified the end of a wrapped line is marked by "\" as the last character on the line.

Only users with SUPER.SUPER privileges (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access are allowed to perform the EXPORT SSHCTL function. The following export files are generated:

File	Description
USER	USER object data
USERPUBK	All public keys of all users
PRIVKEY	KEY object data
KNWNHOST	KNOWNHOST object data
RESTRICT	RESTRICTION-PROFILE object data

```
%export sshctl, subvol $data1.sshexp
OK, all SSHCTL exported to files on $data1.sshexp
%
```

INFO HOST-KEY

The INFO HOST-KEY provides detailed information about the host key that is stored in the HOSTKEY file: name of the hostkey file, type of key, size of key and the key's fingerprints (bubble-babble and MD5).

The command has the following syntax:

```
INFO HOST-KEY
```

All users with SSHCOM access can execute this command.

Example:

```
% info host-key
info host-key
  HOSTKEY-FILE HOSTKEY
  TYPE ssh-dss
  BITS 1024
  PUBLICKEY-FINGERPRINT
    MD5: 23:42:77:e1:20:51:ff:55:e7:4c:7a:c8:71:30:06:93
    BABBLE: xuseb-mofen-sisuh-zogun-cehuz-pomaz-vuzuf-tabup-lodoz-lured-ruxix
%
```

The MD5 fingerprint is logged at SSH2 process startup as well.

The fingerprint information can be used to configure a known host entry on a remote system.

EXPORT HOST-KEY

The EXPORT HOST-KEY command will export the public key part of the host key that is stored in the HOSTKEY file.

The command has the following syntax:

```
EXPORT HOST-KEY, FILE {<GUARDIAN-file-name> | "<OSS-file-name>" | <OSS-file-name> }
```

The individual attributes have the following meaning and syntax:

```
FILE {<GUARDIAN-file-name> | "<OSS-file-name>" | <OSS-file-name> }
```

The name of the Guardian or OSS file that will hold the exported key. A file created in the Guardian name space will be a file with file code 180. If an OSS file name is specified that contains spaces (or commas), then double quotes are required for the attribute value.

All users with SSHCOM access can execute this command.

Example:

```
% export host-key, file $temp.sshtemp.hostkey1
export host-key, file $temp.sshtemp.hostkey1
OK, written public part of host key to file $temp.sshtemp.hostkey1
%
```

The exported file can be used to configure a known host entry on a remote system.

Daemon Mode Commands - Overview

The SSH2 user base is maintained using the following commands. The commands will be discussed in details in the following subsections. Please also see "Database for Daemon Mode" in chapter "The SSH User Database" for an overview of the database content.

- Commands operating on the USER entity:
  - ADD USER: adds a new user to the database.
  - ALTER USER: changes parameters for an existing user.
  - DELETE USER: deletes an existing user.
  - FREEZE USER: freezes a user name, rendering it unable to log on from remote.
  - INFO USER: shows information about a user or a set of users.
  - RENAME USER: renames a user.
  - THAW USER: thaws a user name, making it active again.
- Commands operating on the RESTRICTION-PROFILE entity:
  - ADD RESTRICTION-PROFILE: adds a new restriction profile to the database.
  - ALTER RESTRICTION-PROFILE: changes parameters for an existing restriction profile.
  - DELETE RESTRICTION-PROFILE: deletes an existing restriction profile.
  - INFO RESTRICTION-PROFILE: shows information about a restriction profile or a set of restriction profiles.
  - RENAME RESTRICTION-PROFILE: renames a restriction profile.

Daemon Mode Commands Operating on the USER Entity

ADD USER

The ADD USER command adds a new user to the database and has the following syntax:

```
ADD USER <user-name>
        [,ALLOW-CI yes|no ]
        [,ALLOW-CI-PROGRAM-OVERRIDE yes|no ]
        [,ALLOW-GATEWAY-PORTS yes|no ]
        [,ALLOW-MULTIPLE-REMOTE-HOSTS yes|no ]
        [,ALLOW-PTY yes|no ]
        [,ALLOW-SHELL yes|no ]
        [,ALLOW-TCP-FORWARDING yes|no ]
        [,ALLOWED-AUTHENTICATIONS ( <method>, <method>, ...) | <method> ]
        [,ALLOWED-SUBSYSTEMS ( <subsystem>, <subsystem>, ... ) | <subsystem> ]
        [,CI-COMMAND [ <command> ] ]
        [,CI-PROGRAM [ <filename> | *MENU* | *MENU* <service> [ FORCE ] ] ]
        [,COMMENT <comment> | "<comment containing spaces>" ]
        [,CPU-SET [<cpu> | <cpu-range> | ( <cpu-range-list> ) ] ]
        [,FROZEN]
        [,LIKE <existing-user-name>]
        [,OWNER < system-user-name> | *NONE*]
        [,PRINCIPAL { <user>@<REALM> | *@<REALM> | *@* } ]
        [,PRIORITY -1 | <priority> ]
        [,PTY-SERVER { *DEFAULT* | <process-name> } ]
        [,PUBLICKEY <key-name> { FINGERPRINT <fingerprint-value> |
                                 FILE <filename> } |
                               ( { FINGERPRINT <fingerprint-value> |
                                   FILE <filename> }
                                 [, COMMENT "<comment>"]
                                 [, LIVE-DATE <date-time>]
                                 [, EXPIRE-DATE <date-time>] )
                             ]...
        [,RESTRICTION-PROFILE [<profile-name>] ]
        [,SFTP-CPU-SET [<cpu> | <cpu-range> | ( <cpu-range-list> ) ] ]
        [,SFTP-GUARDIAN-FILESET ( <pattern>, <pattern>, ... ) ]
        [,SFTP-INITIAL-DIRECTORY <directory-path> [LOCKED]]
        [,SFTP-PRIORITY [ <number> ] ]
        [,SFTP-SECURITY ( [<sftp-attr>] [, <sftp-attr>] ... ) ]
        [,SHELL-COMMAND [ <command> ] ]
        [,SHELL-ENVIRONMENT [ <filename> ] ]
        [,SHELL-PROGRAM [ *DEFAULT* | <path> | *MENU* | *MENU* <service> [ FORCE ] ] ]
        [,SYSTEM-USER <system-user-name> | *NONE* ]
```

Only the <user-name> is mandatory in the command, all other fields are optional.

The individual attributes have the following meaning and syntax:

<user-name>

The name of the user to be added. It is not required that this user is a Guardian user name but Guardian user names like ADMIN.JOE or alias names can be used. The important bit here is to be aware that this SSH user name is not used as logon name: The actual Guardian user is defined by the attribute SYSTEM-USER.

It is possible to specify a logon id in double quotes, which allows to execute client commands like ssh 110,23@NonStop.com. But only if SYSTEM-USER is set to "110,23" or the corresponding <group>.<user> value (or an alias with that logon id) the operations on the NonStop server will be executed with logon id 110,23.

It is also possible to have an unconventional SSH logon name different from the system-user name, for instance, ADD USER "super.super,test", system-user super.super, when double quotes are used.

ALLOW-CI

This attribute controls whether a TACL or a specific command interpreter given by CI-PROGRAM should be started upon a shell request of a client that allocated a 6530 pseudo TTY (such as 6530 SSH clients, MR-Win6530, and J6530).

ALLOW-CI-PROGRAM-OVERRIDE

This attribute controls if a user is allowed to override the configured CI-PROGRAM via "tacl -p" or "ci -p" command. If the CI-PROGRAM is set to *DEFAULT*, i.e. command interpreter TACL gets started and ALLOWED-SUBSYSTEMS contains tacl, then this attribute is ignored because a user can start TACL and execute any command interpreter in that way. In this case it is useless to try preventing "tacl -p" commands. The parameter is especially useful in cases where the user does not have tacl as ALLOWED-SUBSYSTEM but needs to be allowed to execute some specific command interpreter or TACL macro. If CI-PROGRAM is configured with a specific command interpreter or macro and ALLOW-CI-PROGRAM-OVERRIDE is set to NO, then a user is restricted to execute the configured CI-PROGRAM and will not get a TACL prompt. Should the ALLOW-CI-PROGRAM-OVERRIDE be YES, then the user can execute a "tacl -p <program>" or a "ci -p <program>" command, thus overriding the program configured in CI-PROGRAM.

ALLOW-GATEWAY-PORTS

This attribute is used to grant or deny gateway ports when port forwarding is initiated by a specific user. If the value of this attribute is NO, then any port forwarding request with SSH option -g will be rejected by SSH2.

ALLOW-MULTIPLE-REMOTE-HOSTS

When set to NO this attribute is used to restrict a user to a maximum of one remote host the user can establish a connection from at any time. The restriction is based on the SSH user configured in the SSH2 database (not the system user). After disconnecting all sessions from one host the user can connect from a different host. All SSH2 processes that access the same SSH2 database share the restriction. If the attribute is set to YES, then a user can establish sessions from different remote hosts at the same time.

ALLOW-PTY

This attribute is used to grant or deny the allocation of a pseudo TTY for a session. The pseudo TTY enables the user to execute full screen interactive applications, such as Emacs or vi.

ALLOW-SHELL

This attribute is used to grant or deny shell access to a user.

ALLOW-TCP-FORWARDING

This attribute is used to grant or deny port forwarding for a user. The value of this user attribute is ignored if the global SSH2 parameter ALLOWTCPFORWARDING is set to FALSE.

ALLOWED-AUTHENTICATIONS

This attribute is used to specify the authentication mechanisms that are allowed for a user. The following authentication methods currently supported by SSH2:

- password: Password authentication facilitating the NonStop system's password authentication mechanism. The password is validated against the SYSTEM-USER's password. Local authentication with password now provides the remote client IP address to system procedure USER_AUTHENTICATE_ if the OS release supports this (H06.26 or later and J06.15 or later).
- publickey: Public key authentication using the PUBLIC-KEYs configured for a user.
- keyboard-interactive: Authentication according to RFC 4256 mapped to the standard GUARDIAN user authentication dialog, verifying the SYSTEM-USER's password, as well as taking care of exceptions, such as password expiry. Local authentication with password now provides the remote client IP address to system procedure USER_AUTHENTICATE_ if the OS release supports this (H06.26 or later and J06.15 or later).
- none: Grants access without authentication. This is useful for users connecting to an application requiring its own authentication, e.g. if you configure a PATHWAY PROGRAM as a CI-PROGRAM.

CAUTION: When specifying ALLOWED-AUTHENTICATIONS (none) user access should be properly locked down to avoid security breaches that bypass any authentication (e.g. by setting SYSTEM-USER *NONE*).

ALLOWED-SUBSYSTEMS

This attribute is used to control access to specific subsystems. <subsystem> is one of the following subsystems provided by SSH2:

- SFTP: The SFTP subsystem allows the user to transfer files with the SFTP transfer protocol.
- TACL: The TACL subsystem provides direct TACL access without requiring OSS on the NonStop server.

CI-COMMAND

This attribute specifies the startup string to be passed to CI-PROGRAM. Specify CI-COMMAND without <command> to reset the attribute to its default (an empty startup string).

CI-COMMAND is ignored if CI-PROGRAM is set to *MENU*.

CI-PROGRAM

Sets the command interpreter to be started on a 6530 pseudo TTY after this user is authenticated. The filename is the name of the command interpreter's object file. It must be a local file name.

If you omit any attribute value, CI-PROGRAM will be reset to its default (TACL).

Startup parameters can be specified for the configured program, which is especially of interest for the program value TELNET (please refer to section "Using TELSERV as Service Provider").

Please note: Specifying startup parameters in addition to the program file name requires double quotes around the CI-PROGRAM attribute value, for example:

ADD USER ...., CI-PROGRAM "TELNET <ip-addr> <port>".

If *MENU* is specified, 6530 shell will be connected to the service menu provided by the STN PTYSERVER. This resembles the functionality of TELSERV, which provides dynamic services, as well as services connecting to static windows. The services offered by the STN PTYSERVER process can be configured using STNCOM.

ALLOW-PTY must be set to YES for this attribute to be accepted for 6530 SSH clients, such as MR-Win6530 or J6530.

If *MENU* is followed by a service or window name, the corresponding service or window is automatically selected. If the service or window does not exist, the STN menu will be displayed.

If the option FORCE is appended, then the user is forced to use the pre-configured STN service or window. In this case, the user will not see the STN menu, even when the configured service or window does not exist.

COMMENT

Enables the input of free text enabling administrators to describe an entity or provide a short explanation of the intended use of the USER entity or, when COMMENT is used for a PUBLICKEY, for the user public key. The whole comment must be enclosed in double quotes if the comment includes spaces. The content will not be used for any processing.

CPU-SET

Defines a set of CPUs used when processes (except SFTPSERV processes) are invoked directly by SSH2 (for SFTPSERV processes the attribute SFTP-CPU-SET is used instead). CPUs are assigned via a round-robin algorithm among all the configured CPUs that are available.

The value can be a CPU number (e.g. 2), a range of CPUs (e.g. 3-4), or a comma-separated list of CPU numbers and CPU ranges, enclosed in parentheses, e.g. (2, 5-7, 9).

The default is to start user processes in the same CPU in which the SSH2 process is running. In this case, the processing load is spread by using multiple SSH2 processes and starting these SSH2 processes in different CPUs).

If no value is specified, the value will be reset to the default. The default is to use the value of SSH2 parameter CPUSET to determine a CPU or, if that is not set, the CPU the SSH2 process is running in is used.

EXPIRE-DATE

This optional attribute of an ssh user's PUBLICKEY entry is used to set the EXPIRE-DATE (not-valid-after date) for the public key. This attribute can only be set if the life-cycle policy for User Public Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to FIXED, then field EXPIRE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to VARIABLE, then every user with partial SSHCOM access can change field EXPIRE-DATE.

FROZEN

If the FROZEN attribute is set, the user is added in the frozen state. If omitted, the user will be added in the thawed state.

LIKE

When specified, the new user record is first initialized with the values taken from the <existing-user-name> user record. Then the new user name and any other attributes specified in the ADD USER command are applied before the new user record is added. If the ADD USER command does not include a SYSTEM-USER attribute, then the new user name is used as SYSTEM-USER as well unless the SSH2 parameter USETEMPLATESYSTEMUSER is true (in that case the new user record will get the value for the SYSTEM-USER attribute from the <existing-user-name> user record).

LIVE-DATE

This optional attribute of an ssh user's PUBLICKEY entry is used to set the LIVE-DATE (not-valid-before date) for the public key. This attribute can only be set if the life-cycle policy for User Public Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to FIXED, then field LIVE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to VARIABLE, then every user with partial SSHCOM access can change field LIVE-DATE.

OWNER

Allow an existing local user to modify all USER records that are configured with that local user as value for USER attribute OWNER. The allowed actions will be the same as defined by PARTIALSSHCOMACCESSUSER/GROUP parameters. The OWNER field for existing USER records will be assumed to be "*NONE*" which means the user that is currently logged in. New USER records will also be set to OWNER "*NONE*" by default unless attribute OWNER is explicitly set to a different value. The owner could be identical to the SYSTEM-USER value, could be "SUPER.SUPER" or the group manager of the user configured in SYSTEM-USER or could be any other local system user.

PRINCIPAL

When Kerberos is implemented on the system, this attribute is used to explicitly specify which Kerberos principal(s) are authorized to logon to this user account using "gssapi-with-mic" authentication. To define an access control list with multiple principals within a single command, the PRINCIPAL attribute can be repeated within a single ADD USER command.

Note: Specifying one or more Kerberos principals using this attribute will override the default Kerberos authorization rule, which implicitly grants access to the Kerberos principal with a matching local account name.

The PRINCIPAL attribute may have the following values:

- <user>@<REALM>
  A fully qualified Kerberos principal name will authorize a specific Kerberos principal to access this user account

- *@<REALM>
  This pattern will authorize any principal in the given REALM to access this user account
- *@*
  This pattern will authorize any principal in any REALM (i.e. anybody with a valid service ticket) to access this user account

Note: Specifying a wildcard pattern as principal is useful when delegating authorization to the resource started for this user (i.e. CI-PROGRAM or SHELL-PROGRAM).

CAUTION: When specifying a wildcard PRINCIPAL, user access should be properly locked down to avoid security breaches in which per-user authorization is bypassed (e.g. by setting SYSTEM-USER *NONE*).

The Kerberos principal name authenticated and authorized during “gssapi-with-mic” authentication will also be displayed in the audit log and thus can be used to correlate the Kerberos principal name with the NonStop user name.

To delete a PRINCIPAL from the access control list, use the DELETE PRINCIPAL attribute.

PRIORITY

All user processes (except SFTPSERV processes) started directly by SSH2 will have the configured priority assigned. Following are the values allowed in this parameter and their meanings:

Value	Meaning
1-199	Use the given priority value
-1	Use the same priority as the SSH2 process starting the process.

Note: SFTPSERV processes will be given priority as specified via the SFTP-PRIORITY attribute.

PTY-SERVER

The value of a specific STN PTY server, Guardian process name, which the user will use.

If a value of *DEFAULT* is specified, the user will use the STN PTY server that is configured via SSH2 parameter PTYSERVER.

PUBLICKEY

This attribute is used to assign one or more public key(s) to a user. Each public key must be given a <key-name> which is unique among all public keys assigned to the current user. The key name will also be displayed in the audit log and thus can be used to determine which public key has been used for logon at a given time.

To add multiple public keys within a single command, the PUBLICKEY attribute can be repeated within a single ADD USER command. There is no limitation to the number of public keys that can be assigned to a user.

Public keys can be added by either specifying a file containing the public key or by specifying the fingerprint of the public key.

To specify a file holding the public key, the key word FILE must be used. The <filename> needs to point to a file holding the public key to be added. For details about the format of the public key file, refer to the chapter entitled "SSH Protocol Reference".

Instead of providing a public key file, it is possible to only provide the fingerprint of the user's public key. In this case, the key word FINGERPRINT must be used, followed by the fingerprint of the user's public key, which should be specified either in MD5 or "bubble-babble" form and enclosed in double-quotes.

Note: Only one of the two key words FILE or FINGERPRINT can be used in a single PUBLICKEY attribute specification.

RESTRICTION-PROFILE

Specifies the name of a RESTRICTION-PROFILE entity. If configured for a user, then the restrictions defined in the RESTRICTION-PROFILE record will be applied for all of a user's incoming and outgoing connections.

SFTP-CPU-SET

Defines a set of CPUs used when SFTPSERV processes are invoked directly by SSH2 (for non-SFTPSERV processes the attribute CPU-SET is used instead). CPUs are assigned via a round-robin algorithm among all the configured CPUs that are available.

The value can be a CPU number (e.g. 2), a range of CPUs (e.g. 3-4), or a comma-separated list of CPU numbers and CPU ranges, enclosed in parentheses, e.g. (2, 5-7, 9).

The default is to start user processes in the same CPU in which the SSH2 process is running. In this case, the processing load is spread by using multiple SSH2 processes and starting these SSH2 processes in different CPUs).

If no value is specified, the value will be reset to the default. The default is to use the value of SSH2 parameter SFTPCPUSET to determine a CPU or, if that is not set, the CPU the SSH2 process is running in is used.

SFTP-GUARDIAN-FILESET

A list of patterns identifying the GUARDIAN systems, volumes, subvolumes, and files the user is allowed to access. Following is the default for this attribute:

```
('\*.$*.*.*)
```

The default enables access (limited by the SFTP-SECURITY attribute) to any GUARDIAN system, volume, subvolume, or file. In each pattern configured with the GUARDIAN file set, the '*' sign is used as a wildcard for any sequence of characters. The '?' sign is used in a pattern as a wildcard for one single character.

SFTP-INITIAL-DIRECTORY

This attribute specifies the initial server-side directory the user will access after establishing the SFTP session. The default value for the initial directory is either the value taken from INITIAL-DIRECTORY when defined in Safeguard or from the Guardian default subvolume of the SYSTEM-USER.

If the option LOCKED is used, a user will not be allowed to leave that path, by issuing a "cd .." command. For example, if a value of "/home/jdoe" is used, only access to directories below is allowed. Access to upper level directories such as "/home" or "/usr" or "/" will not be allowed. Specifying option LOCKED results in a pseudo root visible for the user, i.e. a pwd command will show "/" as current directory.

If a value /G LOCKED is used, then the user can only access Guardian files and no OSS files.

SFTP-PRIORITY

A number specifying the priority of the SFTPSERV processes for this user. Following are the values allowed in this parameter and their meanings:

Value	Meaning
1-199	Use the given priority value
-1	Use the same priority as the SSH2 process starting SFTPSERV

The default value is 100

SFTP-SECURITY

This parameter is comprised of a comma-separated list of allowed operations for the user, with operations enclosed in brackets. The operations allowed are as follows:

- LIST: allows perusal of files
- READ: allows downloading of files to the remote system
- WRITE: allows uploading of files from the remote system
- PURGE: allows deletion of files on the NonStop system

- RENAME: allows renaming of files on the NonStop system
- MKDIR: allows creation of directories on the NonStop system
- RMDIR: allows removal of directories on the NonStop system
- SYMLINK: allows creation of symbolic links on the NonStop system
- ALL: shortcut for all operations
- NONE: shortcut for no operation

Operations can be abbreviated as long as the abbreviation is unambiguous.

Example:

- SFTP-SECURITY (WRITE,LIST)
  - will only allow perusal of files and uploading of files
  - can be abbreviated as SFTP-SECURITY (W,L)

SHELL-COMMAND

This attribute specifies a forced command that is to be executed rather than any command given by an exec request from the SSH client. A forced command allows you to limit shell access to specific tasks or implement additional security measures. SSH2 will retain the command given in the user's exec request, in the SSH_ORIGINAL_COMMAND environment variable, to allow a shell script to analyze and/or execute the original command.

SHELL-ENVIRONMENT

The full OSS file name of a shell script preparing the shell environment for non-login shells (which are started without executing /etc/profile or ~/.profile). The value will be used to set environment variable ENV (see man pages of ksh for information on how the shell processes ENV). The attribute value (shell script) can contain absolute paths but also pre-defined values like $HOME or ~.

Default for this parameter: empty string, i.e. no shell script will be executed that prepares the user environment for non-login shells (which do not execute the standard login scripts). This is relevant for an SCP configuration where the SCP program must be in a directory that is listed in environment variable PATH for getting file transfers using SCP to work.

SHELL-PROGRAM

This attribute specifies the path to the shell program that is to be used to start a shell or execute a command. Specify *DEFAULT* or SHELL-PROGRAM without argument to make SSH2 use the default initial program configured for the assigned SYSTEM-USER (e.g. by the INITIAL-PROGRAM attribute of a SAFEGUARD user).

If *MENU* is specified, the non-6530 session will be connected to a service menu provided by the STN PTYSERVER. This resembles the functionality of TELSERV, providing dynamic services, as well as services connecting to static windows. The services offered by the STN PTYSERVER process can be configured using STNCOM.

If *MENU* is followed by a service or window name, the corresponding service or window is automatically selected. If the service or window does not exist, the STN menu will be displayed.

If the option FORCE is appended, then the user is forced to use the pre-configured STN service or window. In this case the user will not see the STN menu, even when the configured service or window does not exist.

Example for setting up and invoking a non-login shell script (non-interactive) to execute in a ksh shell:

```
% ALTER USER xyx, SHELL-PROGRAM /bin/ksh
```

A ksh shell will be started when the SSH client is invoked.

The second step is to ensure that the PATH variable is set. For non-interactive shells the default scripts do not get executed and the PATH is not defined. For this purpose, SHELL-ENVIRONMENT needs to be set via SSHCOM command:

```
% ALTER USER xyz, SHELL-ENVIRONMENT /home/xyz/myPATH
```

In this example, the script /home/xyz/myPATH contains:

```
export PATH=$PATH:/usr/bin
```

The third step is to create an executable shell script /usr/bin/test-script, for example:

```
echo Entering $0
echo Parameters=\>$*\<
echo ----------------------------------------
echo \$SSH_ORIGINAL_COMMAND:
echo $SSH_ORIGINAL_COMMAND
echo ----------------------------------------
echo Leaving $0
```

Now the actual test is executed by starting an ssh client:

```
C:\WINDOWS>ssh -oPort=15022 xyz@10.0.0.194 test-script  '/home/xyz/repo1/'
xyz@10.0.0.194's password: ...

Entering test-script
Parameters=>/home/xyz/repo1/<
 ----------------------------------------
 $SSH_ORIGINAL_COMMAND:
 test-script  /home/xyz/repo1/
 ----------------------------------------
Leaving test-script
```

SYSTEM-USER

This attribute defines the Guardian user name to which the <user-name> is mapped.

If this attribute is omitted, it is assumed that <user-name> is a valid user on the system.

If *NONE* is specified, the user is not mapped to a system user, causing all channel requests that require a valid system user (e.g. exec, subsystem SFTP) to be rejected. SYSTEM-USER *NONE* is useful to grant anonymous access to services which perform their own authentication (e.g. Pathway applications). When SYSTEM-USER *NONE* is used and CI-PROGRAM or SHELL-PROGRAM are *MENU* and TACL or OSH can be selected from the STN menu, then a logon for TACL or OSS is required.

It is possible to specify the logon id (e.g. 11,23) in double quotes. The logon id will be converted to <group>.<user> before the value for SYSTEM-USER is set.

ALTER USER

The ALTER USER command changes one or more attributes of an existing user and has the following syntax:

```
ALTER USER <user-name>
        [,ALLOW-CI yes|no ]
        [,ALLOW-CI-PROGRAM-OVERRIDE yes|no ]
        [,ALLOW-GATEWAY-PORTS yes|no ]
        [,ALLOW-MULTIPLE-REMOTE-HOSTS yes|no ]
        [,ALLOW-PTY yes|no ]
        [,ALLOW-SHELL yes|no ]
        [,ALLOW-TCP-FORWARDING yes|no ]
        [,ALLOWED-AUTHENTICATIONS ( <method>, <method>, ... ) | <method> ]
        [,ALLOWED-SUBSYSTEMS ( <subsystem>, <subsystem>, ... ) | <subsystem> ]
        [,CI-COMMAND [ <command> ] ]
        [,CI-PROGRAM [ <filename> | *MENU* | *MENU* <service> [ FORCE] ] ]
        [,COMMENT <comment> | "<comment containing spaces>" ]
        [,CPU-SET [<cpu> | <cpu-range> | ( <cpu-range-list> ) ] ]
        [,DELETE PRINCIPAL { <user>@<REALM> | *@<REALM> | *@* | * } ] ...
        [,DELETE PUBLICKEY { <key-name> | * }] ...
        [,OWNER < system-user-name> | *NONE*]
        [,PRINCIPAL { <user>@<REALM> | *@<REALM> | *@* } ] ...
        [,PRIORITY -1 | <priority> ]
        [,PTY-SERVER { *DEFAULT* | <process-name> } ]
        [,PUBLICKEY <key-name> FINGERPRINT <fingerprint-value> |
                               FILE <filename> |
```

```
                               COMMENT "<comment>"] |
                               LIVE-DATE <date-time>] |
                               EXPIRE-DATE <date-time>] |
                               ( [  FINGERPRINT <fingerprint-value>]
                                 [, FILE <filename>]
                                 [, COMMENT "<comment>"]
                                 [, LIVE-DATE <date-time>]
                                 [, EXPIRE-DATE <date-time>] )
                           ]...
        [,RESET { SFTP-INITIAL-DIRECTORY | SYSTEM-USER |
                      SFTP-SECURITY | SFTP-GUARDIAN-FILESET |
                      SFTP-PRIORITY } ]
        [,RESTRICTION-PROFILE [<profile-name>] ]
        [,SFTP-CPU-SET [<cpu> | <cpu-range> | ( <cpu-range-list> ) ] ]
        [,SFTP-GUARDIAN-FILESET ( <pattern>, <pattern>, ... ) ]
        [,SFTP-INITIAL-DIRECTORY <directory-path> [LOCKED]]
        [,SFTP-PRIORITY [ <number> ] ]
        [,SFTP-SECURITY ( [<sftp-attr>] [, <sftp-attr>] ... ) ]
        [,SHELL-COMMAND [ <command> ] ]
        [,SHELL-ENVIRONMENT [ <filename> ]]
        [,SHELL-PROGRAM [ *DEFAULT* | <path> | *MENU* | *MENU* <service> [ FORCE ] ] ]
        [,SYSTEM-USER <system-user-name> | *NONE* ]
```

The <user-name> is mandatory in the command, no wild cards are allowed in the user name. Please see description of <user-name> under the ADD USER command for unconventional names that must be put in double quotes. At least one attribute needs to be specified in the command.

The individual attributes have the following meaning and syntax:

ALLOW-CI

This attribute controls whether a TACL or a specific command interpreter given by CI-PROGRAM should be started upon a shell request of a client that allocated a 6530 pseudo TTY (such as 6530 SSH clients, MR-Win6530, and J6530).

ALLOW-CI-PROGRAM-OVERRIDE

This attribute controls if a user is allowed to override the configured CI-PROGRAM via "tacl -p" or "ci -p" command. If the CI-PROGRAM is set to *DEFAULT*, i.e. command interpreter TACL gets started and ALLOWED-SUBSYSTEMS contains tacl, then this attribute is ignored because a user can start TACL and execute any command interpreter in that way. In this case it is useless to try preventing "tacl -p" commands. The parameter is especially useful in cases where the user does not have tacl as ALLOWED-SUBSYSTEM but needs to be allowed to execute some specific command interpreter or TACL macro. If CI-PROGRAM is configured with a specific command interpreter or macro and ALLOW-CI-PROGRAM-OVERRIDE is set to NO, then a user is restricted to execute the configured CI-PROGRAM and will not get a TACL prompt. Should the ALLOW-CI-PROGRAM-OVERRIDE be YES, then the user can execute a "tacl -p <program>" or a "ci -p <program>" command, thus overriding the program configured in CI-PROGRAM.

ALLOW-GATEWAY-PORTS

This attribute is used to grant or deny gateway ports in the case of port forwarding initiated by a specific user. If the value of this attribute is NO, then any port forwarding request with SSH option "-g" will be rejected by SSH2.

ALLOW-MULTIPLE-REMOTE-HOSTS

When set to NO this attribute is used to restrict a user to a maximum of one remote host the user can establish a connection from at any time. The restriction is based on the SSH user configured in the SSH2 database (not the system user). After disconnecting all sessions from one host the user can connect from a different host. All SSH2 processes that access the same SSH2 database share the restriction. If the attribute is set to YES, then a user can establish sessions from different remote hosts at the same time.

ALLOW-PTY

This attribute is used to grant or deny the ability to allocate a pseudo TTY for a session. The pseudo TTY enables the user to execute full screen interactive applications, such as Emacs or vi.

ALLOW-SHELL

This attribute is used to grant or deny shell access to the user.

ALLOW-TCP-FORWARDING

This attribute is used to grant or deny port forwarding for a user. The value of this user attribute is ignored if the global SSH2 parameter ALLOWTCPFORWARDING is set to FALSE.

ALLOWED-AUTHENTICATIONS

This attribute is used to specify the authentication mechanisms that are allowed for this user. <method> is one of the following authentication methods currently supported by SSH2:

- password: Password authentication facilitating the NonStop system's password authentication mechanism. The password is validated against the SYSTEM-USER's password. Local authentication with password now provides the remote client IP address to system procedure USER_AUTHENTICATE_ if the OS release supports this (H06.26 or later and J06.15 or later).
- publickey: Public key authentication using the PUBLIC-KEYs configured for this user.
- keyboard-interactive: Authentication according to RFC 4256 mapped to the standard GUARDIAN user authentication dialog verifying the SYSTEM-USER's password, as well as taking care of exceptions such as password expiry. Local authentication with password now provides the remote client IP address to system procedure USER_AUTHENTICATE_ if the OS release supports this (H06.26 or later and J06.15 or later).
- none: Grants access without authentication. This is useful for users connecting to an application requiring its own authentication, e.g. if you configure a PATHWAY PROGRAM as CI-PROGRAM.

CAUTION: When specifying ALLOWED-AUTHENTICATIONS (none) user access should be properly locked down to avoid security breaches that bypass any authentication (e.g. by setting SYSTEM-USER *NONE*).

ALLOWED-SUBSYSTEMS

This attribute is used to control access to specific subsystems. <subsystem> is one of the following subsystems provided by SSH2:

- SFTP: The SFTP subsystem allows the user to transfer files with the SFTP transfer protocol.
- TACL: The TACL subsystem provides direct TACL access without requiring OSS on the NonStop server.

CI-COMMAND

This attribute specifies the startup string to be passed to CI-PROGRAM. Specify CI-COMMAND without <command> to reset the attribute to its default (empty startup string).

CI-COMMAND is ignored if CI-PROGRAM is set to *MENU*.

CI-PROGRAM

Sets the command interpreter to be started on a 6530 pseudo TTY after the user is authenticated. In this case, filename is the name of the command interpreter’s object file. It must be a local file name.

If you omit any attribute value, CI-PROGRAM will be reset to its default (TACL).

Startup parameters can be specified for the configured program, which is especially of interest for the program value TELNET (please refer to section "Using TELSERV as Service Provider").

Please note: Specifying startup parameters in addition to the program file name requires double quotes around the CI-PROGRAM attribute value, for example:

ALTER USER ...., CI-PROGRAM "TELNET <ip-addr> <port>".

If *MENU* is specified, 6530 shell will be connected to the service menu provided by the STN PTYSERVER. This resembles the functionality of TELSERV, which provides dynamic services, as well as services connecting to static windows. The services offered by the STN PTYSERVER process can be configured using STNCOM.

ALLOW-PTY must be set to YES for this attribute to be accepted for 6530 SSH clients, such as MR-Win6530 or J6530.

If *MENU* is followed by a service or window name, the corresponding service or window is automatically selected. If the service or window does not exist, the STN menu will be displayed.

If the option FORCE is appended, then the user is forced to use the pre-configured STN service or window. In this case the user will not see the STN menu, even when the configured service or window does not exist.

COMMENT

Enables administrators to input free text that describes an entity or provides a short explanation of the intended use of the USER entity or, when COMMENT is used for a PUBLICKEY, for the user public key. The entire comment must be enclosed in double quotes if the comment includes spaces. The content will not be used for any processing.

CPU-SET

Defines a set of CPUs used when processes (except SFTPSERV processes) are invoked directly by SSH2 (for SFTPSERV processes the attribute SFTP-CPU-SET is used instead). CPUs are assigned via a round-robin algorithm among all the configured CPUs that are available.

The value can be a CPU number (e.g. 2), a range of CPUs (e.g. 3-4), or a comma-separated list of CPU numbers and CPU ranges, enclosed in parentheses, e.g. (2, 5-7, 9).

The default is to start user processes in the same CPU in which the SSH2 process is running. In this case, the processing load is spread by using multiple SSH2 processes and starting these SSH2 processes in different CPUs).

If no value is specified, the value will be reset to the default. The default is to use the value of SSH2 parameter CPUSET to determine a CPU or, if that is not set, the CPU the SSH2 process is running in is used.

DELETE PRINCIPAL

Deletes the principal name specified by <user>@<REALM>, a pattern or all principal names from the list of principal names defined for the user. If more than one valid principal name is to be deleted by name, then there must be one DELETE PRINCIPAL <user>@<REALM> attribute for each principal name. If *@<REALM> is specified the entry *@<REALM> is removed and not all principal names ending in <REALM>. Similarly, when *@* is specified the principal entry *@* is removed from the list of principals. If all entries need to be removed from the user's list of principals the wildcard * can be used, i.e. DELETE PRINCIPAL *.

DELETE PUBLICKEY

This attribute deletes the public key identified by <key-name> or all public keys of the user when wildcard * is specified.

EXPIRE-DATE

This optional attribute of an ssh user’s PUBLICKEY entry is used to set the EXPIRE-DATE (not-valid-after date) for the public key. This attribute can only be set if the life-cycle policy for User Public Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to FIXED, then field EXPIRE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to VARIABLE, then every user with partial SSHCOM access can change field EXPIRE-DATE.

LIVE-DATE

This optional attribute of an ssh user’s PUBLICKEY entry is used to set the LIVE-DATE (not-valid-before date) for the public key. This attribute can only be set if the life-cycle policy for User Public Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to FIXED, then field LIVE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPUBLICUSERKEY is set to VARIABLE, then every user with partial SSHCOM access can change field LIVE-DATE.

OWNER

Similar to the Safeguard USER/ALIAS field OWNER and to base new access rules on that field. This allows an existing local user to modify all USER records that are configured with that local user as value for new USER attribute OWNER. The allowed actions will be the same as defined by PARTIALSSHCOMACCESSUSER/GROUP parameters. The OWNER field for existing USER records will be assumed to be *NONE*. New USER records will be set to OWNER *NONE* by default unless attribute OWNER is explicitly set to a different value. The owner could be identical to the SYSTEM-USER value, could be SUPER.SUPER or the group manager of the user configured in SYSTEM-USER or could be any other local system user.

PRINCIPAL

This attribute is used to explicitly specify which Kerberos principal(s) are authorized to logon to this user account using “gssapi-with-mic” authentication. To define an access control list with multiple principals within a single command, the PRINCIPAL attribute can be repeated within a single ALTER USER command.

Note: Specifying one or more Kerberos principals using this attribute will override the default Kerberos authorization rule, which implicitly grants access to the Kerberos principal with a matching local account name.

The PRINCIPAL attribute may have the following values:

- <user>@<REALM>
  A fully qualified Kerberos principal name will authorize a specific Kerberos principal to access this user account
- *@<REALM>
  This pattern will authorize any principal in the given REALM to access this user account
- *@*
  This pattern will authorize any principal in any REALM (i.e. anybody with a valid service ticket) to access this user account

---

Note: Specifying a wildcard pattern as principal is useful when delegating authorization to the resource started for this user (i.e. CI-PROGRAM or SHELL-PROGRAM).

CAUTION: When specifying a wildcard PRINCIPAL, user access should be properly locked down to avoid security breaches in which per-user authorization is bypassed (e.g. by setting SYSTEM-USER *NONE*).

The Kerberos principal name authenticated and authorized during “gssapi-with-mic” authentication will also be displayed in the audit log and thus can be used to correlate the Kerberos principal name with the NonStop user name.

To delete a PRINCIPAL from the access control list, use the DELETE PRINCIPAL attribute.

PRIORITY

All user processes (except SFTPSERV processes) started directly by SSH2 will have the configured priority assigned. Following are the values allowed in this parameter and their meanings:

Value	Meaning
1-199	Use the given priority value
-1	Use the same priority as the SSH2 process starting the process.

Note: SFTPSERV processes will be prioritized as specified via the SFTP-PRIORITY attribute.

PTY-SERVER

The value of a specific STN PTY server, Guardian process name, which the user will use.

If a value of *DEFAULT* is specified, the user will use the STN PTY server that is configured via SSH2 parameter PTYSERVER.

PUBLICKEY

This attribute is used to add or alter a public key with the provided <key-name>. For details on the syntax of that attribute, please see the "ADD USER" command.

To delete a specific public key for a user use the DELETE PUBLICKEY <key-name> attribute syntax. To delete all public keys for a user, use the DELETE PUBLICKEY * attribute syntax.

Both the PUBLICKEY and the DELETE PUBLICKEY attributes can be repeated multiple times within a single ALTER USER command.

RESET

This option is used to reset an attribute of the current user to the default value. For each attribute that should be reset, there must be a separate occurrence of the RESET option. An attempt to set and reset an attribute will result in an error message.

The following attributes can be reset:

- SFTP-INITIAL-DIRECTORY
- SYSTEM-USER
- SFTP-SECURITY
- SFTP-PRIORITY
- SFTP-GUARDIAN-FILESET

RESTRICTION-PROFILE

Specifies the name of a RESTRICTION-PROFILE entity. If configured for a user, then the restrictions defined in the RESTRICTION-PROFILE record will be applied for all incoming and outgoing connections related to the user.

SFTP-CPU-SET

Defines a set of CPUs used when SFTPSERV processes are invoked directly by SSH2 (for non-SFTPSERV processes the attribute CPU-SET is used instead). CPUs are assigned via a round-robin algorithm among all the configured CPUs that are available.

The value can be a CPU number (e.g. 2), a range of CPUs (e.g. 3-4), or a comma-separated list of CPU numbers and CPU ranges, enclosed in parentheses, e.g. (2, 5-7, 9).

The default is to start user processes in the same CPU in which the SSH2 process is running. In this case, the processing load is spread by using multiple SSH2 processes and starting these SSH2 processes in different CPUs).

If no value is specified, the value will be reset to the default. The default is to use the value of SSH2 parameter SFTPCPUSET to determine a CPU or, if that is not set, the CPU the SSH2 process is running in is used.

SFTP-GUARDIAN-FILESET

A list of patterns identifying the GUARDIAN systems, volumes, subvolumes and files the user is allowed to access. The default for this attribute is as follows:

```
('\*.$*.*.*)
```

This enables access (limited by the SFTP-SECURITY attribute) to any GUARDIAN system, volume, subvolume, or file. In each pattern configured with the GUARDIAN file set, the '*' sign is used as a wildcard for any sequence of characters. The '?' sign is used in a pattern as a wildcard for one single character.

SFTP-INITIAL-DIRECTORY

This attribute specifies the initial server-side directory the user will access after establishing the SFTP session. The default value for the initial directory is either the value taken from INITIAL-DIRECTORY when defined in Safeguard or from the Guardian default subvolume of the SYSTEM-USER.

If the option LOCKED is used, a user will not be allowed to leave that path, by issuing a "cd .." command. For example, if a value of "/home/jdoe" is used, only access to directories below is allowed. Access to upper level directories such as

"/home" or "/usr" or "/" will not be allowed. Specifying option LOCKED results in a pseudo root visible for the user, i.e. a pwd command will show "/" as current directory.

If a value /G LOCKED is used, then the user can only access Guardian files and no OSS files.

SFTP-PRIORITY

A number specifying the priority of the SFTPSERV processes for this user. Following are the meanings of the values allowed for this parameter:

Value	Meaning
1-199	Use the given priority value
-1	Use the same priority as the SSH2 process starting SFTPSERV

The default value is 100

SFTP-SECURITY

This parameter is comprised of a comma-separated list of allowed operations for the user, with operations enclosed in brackets. The following operations are available:

- LIST: allows perusal of files
- READ: allows downloading of files to the remote system
- WRITE: allows uploading of files from the remote system
- PURGE: allows deletion of files on the NonStop system
- RENAME: allows renaming of files on the NonStop system
- MKDIR: allows creation of directories on the NonStop system
- RMDIR: allows removal of directories on the NonStop system
- SYMLINK: allows creation of symbolic links on the NonStop system
- ALL: shortcut for all operations
- NONE: shortcut for no operation

Operations can be abbreviated as long as the abbreviation is unambiguous.

Example:

- SFTP-SECURITY (WRITE,LIST)
    - will only allow perusal of files and uploading of files
    - can be abbreviated as SFTP-SECURITY (W,L)

SHELL-COMMAND

This attribute specifies a forced command that is to be executed rather than any command given by an exec request from the SSH client. A forced command allows you to limit shell access to specific tasks or implement additional security measures. SSH2 will retain commands given in the user's exec request, in the SSH_ORIGINAL_COMMAND environment variable, to allow a shell script to analyze and/or execute the original command.

SHELL-ENVIRONMENT

The full OSS file name of a shell script preparing the shell environment for non-login shells (which are started without executing /etc/profile or ~/.profile). The value will be used to set environment variable ENV (see man pages of ksh for information on how the shell processes ENV). The attribute value (shell script) can contain absolute paths but also pre-defined values like $HOME or ~.

Default for this parameter: empty string, i.e. no shell script will be executed that prepares the user environment for non-login shells (which do not execute the standard login scripts). This is relevant for an SCP configuration where the SCP program must be in a directory that is listed in environment variable PATH for getting file transfers using SCP to work.

SHELL-PROGRAM

This attribute specifies the path to the shell program to be used to start a shell or execute a command. Specify *DEFAULT* or SHELL-PROGRAM without argument to make SSH2 use the default initial program configured for the assigned SYSTEM-USER (e.g. by the INITIAL-PROGRAM attribute of a SAFEGUARD user).

If *MENU* is specified, the non-6530 session will be connected to a service menu provided by the STN PTYSERVER. This resembles the functionality of TELSERV, providing dynamic services, as well as services connecting to static windows. The services offered by the STN PTYSERVER process can be configured using STNCOM.

If *MENU* is followed by a service or window name, the corresponding service or window is automatically selected. If the service or window does not exist, the STN menu will be displayed.

If the option FORCE is appended, then the user is forced to use the pre-configured STN service or window. In this case the user will not see the STN menu, even when the configured service or window does not exist.

SYSTEM-USER

This attribute defines the Guardian user name to which the <user-name> is mapped.

If this attribute is omitted, it is assumed that <user-name> is a valid user on the system. I.e. the <user-name> value is used for attribute SYSTEM-USER in this case.

If *NONE* is specified, the user is not mapped to a system user, causing all channel requests that require a valid system user (e.g. exec, subsystem SFTP) to be rejected. SYSTEM-USER *NONE* is useful to grant anonymous access to services which perform their own authentication (e.g. Pathway applications). When SYSTEM-USER *NONE* is used and CI-PROGRAM or SHELL-PROGRAM are *MENU* and TACL or OSH can be selected from the STN menu, then a logon for TACL or OSS is required.

It is possible to specify the logon id (e.g. 11,23) in double quotes. The logon id will be converted to <group>.<user> before the value for SYSTEM-USER is set.

DELETE USER

The DELETE USER command deletes a user from the database and has the following syntax:

```
DELETE USER <user-name>
```

The <user-name> is mandatory in the command, and no wild cards are allowed in the user name. Please see description of <user-name> under the ADD USER command for unconventional names that must be put in double quotes.

FREEZE USER

The FREEZE USER command freezes a user and has the following syntax:

```
FREEZE USER <user-name>
```

The <user-name> is mandatory in the command, and no wild cards are allowed in the user name. A frozen user cannot log on from a remote system. Please see description of <user-name> under the ADD USER command for unconventional names that must be put in double quotes.

INFO USER

The INFO USER command displays information about a single user or a set of users and has the following syntax:

```
INFO USER {<user-name> | <user-name-prefix>* | *} [, DETAIL]
```

At least one of <user-name>, <user-name-prefix>* or '*' is mandatory in the command. If <user-name-prefix> followed by an asterisk is specified, the user records are displayed when the first part of the user name matches the specified prefix. If a '*' is used, information for all users will be displayed. Otherwise, information for a single user will be displayed.

For unconventional user names which must be put in in double quotes, please see the <user-name> description under ADD USER.

If used without the DETAIL modifier, INFO USER will provide a brief summary for each user displayed. The following is an example of the output of INFO USER:

```
% info user us1
info user us1

USER                KEYS SYSTEM-USER       LAST-MODIFIED LAST-LOGON    STATUS
us1                    2 ulrich            20Apr12,16:00 20Apr12,16:02 THAWED
%
```

If used with the DETAIL modifier, INFO USER will provide some detailed information about each user displayed. The following is an example of the output of INFO USER, DETAIL:

```
% info user us1, detail
info user us1, detail

USER                KEYS SYSTEM-USER       LAST-MODIFIED LAST-LOGON    STATUS
us1                    2 ulrich            20Apr12,16:07 20Apr12,16:02 THAWED

  USER us1
  COMMENT *NONE*
  ALLOWED-AUTHENTICATIONS (password,publickey,keyboard-interactive)
  OWNER *NONE*

  PUBLICKEY k1
    COMMENT used for file transfer from node linux-dev
    MD5    6b:88:75:78:7e:90:bb:7c:eb:0d:94:64:79:07:1f:bd
    BABBLE xegop-hyvik-fucud-tubon-nuvin-pugeg-kovac-vipif-vunym-peset-zyxyx
    CREATION-DATE 20Apr12,15:05
    LIVE-DATE *NONE*
    EXPIRE-DATE *NONE*
    LIFE-CYCLE-STATE LIVE
    LAST-MODIFIED 20Apr12,16:07
    LAST-USAGE *NONE*

  PUBLICKEY testkey3
    COMMENT
    MD5    9e:67:60:36:e0:a4:88:ac:19:f1:39:61:19:0e:88:76
    BABBLE xezaz-fimuf-gacoz-rorid-zutol-cezuc-pygyf-fypes-ponih-lynol-zaxix
    CREATION-DATE 20Apr12,16:00
    LIVE-DATE *NONE*
    EXPIRE-DATE *NONE*
    LIFE-CYCLE-STATE LIVE
    LAST-MODIFIED 20Apr12,16:00
    LAST-USAGE 20Apr12,16:02

  SYSTEM-USER ulrich

  ALLOW-SHELL YES
  SHELL-PROGRAM *DEFAULT*
  SHELL-COMMAND *NONE*
  SHELL-ENVIRONMENT *NONE*
  ALLOW-CI YES
  CI-PROGRAM *DEFAULT*
  CI-COMMAND *NONE*
  ALLOW-PTY YES
  PTY-SERVER $PTY01
  ALLOW-TCP-FORWARDING YES
  ALLOWED-SUBSYSTEMS (sftp,tacl)
  ALLOW-GATEWAY-PORTS YES
  ALLOW-MULTIPLE-REMOTE-HOSTS YES
  RESTRICTION-PROFILE *NONE*
```

```
  PRIORITY -1
  CPU-SET *DEFAULT*

  SFTP-INITIAL-DIRECTORY /G LOCKED
  SFTP-GUARDIAN-FILESET ($temp.us*.*, $us.*.*)
  SFTP-SECURITY (read,write,purge,rename,list,mkdir,rmdir,symlink)
  SFTP-PRIORITY 100
  SFTP-CPU-SET *DEFAULT*

  LAST-LOGON 20Apr12,16:02
  LAST-UNSUCCESSFUL-ATTEMPT *NONE*
  LAST-AUTH-METHOD publickey
  LAST-PUBLICKEY testkey3
  LAST-IP-ADDRESS fe80::a00:8eff:fe00:d14e
  LAST-MODIFIED 20Apr12,16:07
  STATUS THAWED
%
```

Following are the specific fields output by INFO USER and their meaning:

STATUS

Displays whether the user is in a FROZEN or THAWED state.

PUBLICKEY

This field displays fingerprints of the public keys associated with a specific user. For each public key, the name and associated fingerprints are displayed. The last modification and last usage timestamp are also displayed for each public key.

LAST-LOGON

The timestamp of the last successful logon of the user.

Note: For user super.super, the LAST-LOGON timestamp will be updated whenever any user process is started; i.e. the update occurs also when users other than super.super log on.

LAST-UNSUCCESSFUL-ATTEMPT

The timestamp of the last unsuccessful authentication attempt of that user.

LAST-AUTH-METHOD

The last authentication method used for last logon.

LAST-PUBLICKEY

The name of the last public key used for publickey authentication of an incoming ssh connection.

LAST-IP-ADDRESS

The IP address from which the user last connected.

LAST-MODIFIED

The timestamp of the last modification of the user attributes. "User attributes" in that context are attributes that can be changed with the ALTER command.

Note: any attributes not listed above are explained in the “ADD USER” section.

RENAME USER

The RENAME USER command renames a user and has the following syntax:

```
RENAME USER <old-user-name>, <new-user-name>
```

Both <old-user-name> and <new-user-name> are mandatory in the command; no wild cards are allowed in either one.

Please see description of <user-name> under the ADD USER command for unconventional names that must be put in double quotes.

THAW USER

The THAW USER command thaws a user and has the following syntax:

```
THAW USER <user-name>
```

The <user-name> is mandatory in the command, no wild cards are allowed in the user name. A thawed user can log on from a remote system and execute commands. Please see description of <user-name> under the ADD USER command for unconventional names that must be put in double quotes.

Daemon Mode Commands Operating on the RESTRICTION-PROFILE Entity

ADD RESTRICTION-PROFILE

The ADD RESTRICTION-PROFILE command adds a new restriction profile to the database and has the following syntax:

```
ADD RESTRICTION-PROFILE <profile-name>
 [,LIKE <existing-restriction-profile-name>]
 [,COMMENT <comment> | "<comment containing spaces>" ]
 [,CONNECT-FROM <host-pattern> | ( <host-pattern>, <host-pattern, ... ) ]
 [,CONNECT-TO <host-ports> | ( <host-ports>, <host-ports>, ... ) ]
 [,PERMIT-LISTEN <host-ports> | ( <host-ports>, <host-ports>, ... ) ]
 [,PERMIT-OPEN <host-ports> | ( <host-ports>, <host-ports>, ... ) ]
 [,FORWARD-FROM <host-pattern> | ( <host-pattern>, <host-pattern, ... ) ]
```

Only the <profile-name> is mandatory in the command, all other fields are optional. The individual attributes have the following meaning and syntax:

<profile-name>

The name of the restriction profile to be added.

<comment>

A comment describing the restriction profile. If the comment contains spaces, it must be enclosed in double quotes.

<host-pattern>

One or more patterns used to match addresses or names of hosts. Wildcard characters '*' (any number of characters) and '?' (one character) are allowed. The '~' is supported for expressing negation.

<host-ports>

Specifies a pair of host addresses or name and port ranges separated by a colon. A port range can be either one port, one port range or a list of port ranges separated by '+' and enclosed in brackets.

COMMENT

Enables users to enter free text to describe the entity or provide a short explanation of the intended use of the entity. The whole comment text must be enclosed in double quotes if the comment includes spaces. The content will not be used for any processing.

CONNECT-FROM

The attribute CONNECT-FROM restricts the host systems a user can connect from. Whenever an incoming connection for the user is accepted, the CONNECT-FROM restrictions are applied.

The value can be one host pattern or a list of patterns used to match the address or name of the client system connecting SSH2 on NonStop™ server. The format of each pattern and the pattern matching done is the same as in OpenSSH for parameter from=. If a list is specified, it must be enclosed in parentheses.

One pattern represents a host name or its IP address and can include wildcard characters '*' (matching any number of characters) and '?' (matching exactly one character). A pattern may be prefixed by '~' indicating negation, that is, if the matching pattern is preceded by a tilde, the incoming connection will be rejected.

Examples for valid CONNECT-FROM values include:

```
103.10.0.37
dev*
(34.45.56.*, ~34.45.56.12)
(201.30.*.*, tandem1, 120.10.20.?, ~ 120.10.20.7)
```

CONNECT-TO

The CONNECT-TO attribute restricts user access, allowing user-initiated outgoing connections only to the configured host/port combinations. The CONNECT-TO restrictions are applied whenever the user tries to connect via SSH2 using the SSH, SSHOSS, SFTP and SFTPOSS clients.

The value for this attribute can be one host/port range or a list of host/ port ranges. A comma-separated list must be enclosed in parentheses.

Each host/port range is a pair of host and port range, separated by a colon, <host>:<port-range>. A port range can be a single port, a single port range or a list of ports and port ranges separated by + and enclosed in brackets.

Examples for valid values for CONNECT-TO include:

```
103.10.0.47:22
1.2.3.4:1025-1999
yourhost.domain.com:[2013]
abc.domain.com:[2013-2100]
(xyz.domain.com:[22 + 2013-2100 + 5000-5099], 4.5.6.7:[300-301 + 5555])
```

FORWARD-FROM

The attribute FORWARD-FROM restricts a user’s ability to do port forwarding. It restricts the set of hosts that can use forwarding tunnels opened by a specific user.

The value can be one host pattern or a list of patterns used to match the address or name of the client system connecting SSH2 on a NonStop™ server.

Please see the description for the CONNECT-FROM attribute for examples.

LIKE

When specified, the new restriction profile record is first initialized with the values taken from the <existing-restriction-profile-name> restriction profile record. Then the new restriction profile name and any other attributes specified in the ADD RESTRICTION-PROFILE command are applied before the new restriction profile record is added.

PERMIT-LISTEN

The PERMIT-LISTEN attribute restricts a user’s ability to do port forwarding. Only the configured ports are allowed for listening on the host opening the forwarding tunnel.

The configuration requires the specification of a host and a port range, but for PERMIT-LISTEN the "host" must either be 0.0.0.0 (indicating gateway ports to follow after the ':') or 127.0.0.1 (indicating non-gateway ports to follow).

PERMIT-OPEN

The PERMIT-OPEN attribute restricts a user’s ability to do port forwarding.

Only the configured host/port combinations are allowed for <targethost> and <targetport> when port forwarding is specified, such as in the following example:

```
ssh -L <localport>:<targethost>:<targetport> <user>@<host>
ssh -R <remoteport>:<targethost>:<targetport> <user>@<host>
```

The PERMIT-OPEN attribute corresponds to the OpenSSH parameter permitopen=.

If localhost or 127.0.0.1 is specified as <targethost>, then the specified <host> is used for restriction checking.

The PERMIT-OPEN restrictions are applied whenever the user tries to establish a local port forwarding channel via SSH2 using the SSH and SSHOSS clients.

For more information regarding format and examples of the attribute value please see the CONNECT-TO attribute section. The format of values for PERMIT-OPEN and CONNECT-TO is the same. The values are just interpreted differently.

ALTER RESTRICTION-PROFILE

The ALTER RESTRICTION-PROFILE command changes one or more attributes of an existing restriction profile and has the following syntax:

```
ALTER RESTRICTION-PROFILE <profile-name>
 [,COMMENT <comment> | "<comment containing spaces>" ]
 [,CONNECT-FROM <host-pattern> | ( <host-pattern>, <host-pattern, ... ) ]
 [,CONNECT-TO <host-ports> | ( <host-ports>, <host-ports>, ... ) ]
 [,PERMIT-LISTEN <host-ports> | ( <host-ports>, <host-ports>, ... ) ]
 [,PERMIT-OPEN <host-ports> | ( <host-ports>, <host-ports>, ... ) ]
 [,FORWARD-FROM <host-pattern> | ( <host-pattern>, <host-pattern, ... ) ]
```

The <profile-name> is mandatory in the command, and no wild cards are allowed in the profile name. At least one attribute needs to be specified in the command.

The individual attributes have the following meaning and syntax:

<profile-name>

The name of the restriction profile to be altered.

<comment>

A comment describing the restriction profile. If the comment contains spaces, it must be enclosed in double quotes.

<host-pattern>

One or more patterns used to match addresses or names of hosts. Wildcard characters '*' (any number of characters) and '?' (one character) are allowed. The '~' is supported for expressing negation.

<host-ports>

Specifies a pair of host addresses or names and port ranges, separated by a colon. A port range can be either one port, one port range or a list of port ranges separated by '+' and enclosed in brackets.

COMMENT

Enables users to enter free text to describe the entity or provide a short explanation of the intended use of the entity. All comment text must be enclosed in double quotes if the comment includes spaces.

The content will not be used for any processing.

CONNECT-FROM

The attribute CONNECT-FROM restricts which host systems a user can connect from. Whenever an incoming connection for the user is accepted, the CONNECT-FROM restrictions are applied.

The value can be one host pattern or a list of patterns used to match the address or name of the client system connecting to SSH2 on the NonStop server. The format of each pattern and the pattern matching done is the same as in OpenSSH for parameter from=.

If a list is specified, it must be enclosed in parentheses.

One pattern represents a host name or its IP address and can include wildcard characters '*' (matching any number of characters) and '?' (matching exactly one character). A pattern may be prefixed by '~' indicating negation, that is, if the matching pattern is preceded by a tilde, the incoming connection will be rejected.

Examples for valid CONNECT-FROM values include:

```
103.10.0.37
dev*
(34.45.56.*, ~34.45.56.12)
(201.30.*.*, tandem1, 120.10.20.?, ~ 120.10.20.7)
```

CONNECT-TO

The CONNECT-TO attribute restricts a user’s outgoing connections to configured host/port combinations. The CONNECT-TO restrictions are applied whenever the user tries to connect via SSH2 using SSH, SSHOSS, SFTP and SFTPOSS clients.

The value for this attribute can be one host/port range or a list of host/port ranges. A comma-separated list must be enclosed in parentheses.

Each host/port range is a pair of host and port ranges, separated by a colon as follows: <host>:<port-range>. A port range can be a single port, a single port range or a list of ports and port ranges separated by + and enclosed in brackets.

Examples of valid values for CONNECT-TO include:

```
103.10.0.47:22
1.2.3.4:1025-1999
yourhost.domain.com:[2013]
abc.domain.com:[2013-2100]
(xyz.domain.com:[22 + 2013-2100 + 5000-5099], 4.5.6.7:[300-301 + 5555])
```

FORWARD-FROM

The FORWARD-FROM attribute restricts a user’s ability to do port forwarding, enabling only a specified set of hosts to use forwarding tunnels opened by a given user.

The value can be one host pattern or a list of patterns used to match the address or name of the client system connecting SSH2 on a NonStop server.

Please see the section on the CONNECT-FROM attribute for examples.

PERMIT-LISTEN

The PERMIT-LISTEN attribute restricts a user’s ability to do port forwarding, enabling only a specified set of hosts to use forwarding tunnels opened by a given user. Only the configured ports are allowed for listening on the host opening the forwarding tunnel.

The configuration requires the specification of a host and a port range, but for PERMIT-LISTEN the "host" must either be 0.0.0.0 (indicating gateway ports to follow after the ':') or 127.0.0.1 (indicating non-gateway ports to follow).

PERMIT-OPEN

The PERMIT-OPEN attribute limits a user’s ability to do port forwarding to only specific host/port combinations. . Configurations are allowed for <targethost> and <targetport> when port forwarding is specified as follows:

```
ssh -L <localport>:<targethost>:<targetport> <user>@<host>
ssh -R <remoteport>:<targethost>:<targetport> <user>@<host>
```

The PERMIT-OPEN attribute corresponds to the OpenSSH parameter permitopen=.

If localhost or 127.0.0.1 is specified as <targethost>, then the specified <host> is used for restriction checking.

The PERMIT-OPEN restrictions are applied whenever the user tries to establish a local port forwarding channel via SSH2 using the SSH and SSHOSS clients.

For formats and examples of the attribute value, please see the CONNECT-TO section. The format of values for PERMIT-OPEN and CONNECT-TO are the same. The values are just interpreted differently.

DELETE RESTRICTION-PROFILE

The DELETE RESTRICTION-PROFILE command deletes a user from the database and has the following syntax:

```
DELETE RESTRICTION-PROFILE <profile-name>
```

The <profile-name> is mandatory in the command, and no wild cards are allowed in the profile name.

INFO RESTRICTION-PROFILE

The INFO RESTRICTION-PROFILE command displays information about a single restriction profile or a set of restriction profiles and has the following syntax:

```
INFO RESTRICTION-PROFILE {<profile-name> | <profile-name-prefix>* | *} [, DETAIL]
```

At least one of <profile-name>, <profile-name-prefix>* or ‘*’ is mandatory in the command. If <profile-name-prefix> followed by an asterisk is specified, the restriction profile records are displayed where the first part of the profile name matches the specified prefix. If a ‘*’ is used, information for all users will be displayed. Otherwise, information for a single user will be displayed.

RENAME RESTRICTION-PROFILE

The RENAME RESTRICTION-PROFILE command renames a restriction profile and has the following syntax:

```
RENAME RESTRICTION-PROFILE <old-profile-name>, <new-profile-name>
```

Both <old-profile-name> and <new-profile-name> are mandatory in the command; no wild cards are allowed in either one.

If the restriction profile <old-profile-name> is in use, that is, if user entries have the RESTRICTION-PROFILE attribute set to the specified <old-profile-name>, the renaming of the restriction profile will be rejected.

Client Mode Commands - Overview

The SSH2 user base is maintained using the following commands:

- Commands operating on the KEY, PASSWORD, and KNOWNHOST entity:
    - ASSUME USER: sets a default user for the following commands.
    - INFO SYSTEM-USER: Displays KEY, PASSWORD, KNOWNHOST information for a specified system user.
- Commands operating on the KEY entity:
    - ALTER KEY: changes properties of a key.
    - DELETE KEY: deletes a key.
    - EXPORT KEY: exports a key into a file. The command supports exporting the public part only as well as exporting the full private key.
    - FREEZE KEY: freezes a key, rendering it inactive.
    - GENERATE KEY: generates a new key and places it into the database.
    - IMPORT KEY: imports a key from a file and places it into the database.
    - INFO KEY: shows information about a key or a set of keys.
    - RENAME KEY: renames a key.
    - THAW KEY: thaws a key, making it active again.
- Commands operating on the PASSWORD entity:
    - ADD PASSWORD: adds a new password to the database.
    - ALTER PASSWORD: changes a password.
    - DELETE PASSWORD: deletes a password.
    - FREEZE PASSWORD: freezes a password, rendering it inactive.
    - INFO PASSWORD: shows information about a key or a set of keys.
    - THAW PASSWORD: thaws a password, making it active again.
- Commands operating on the KNOWNHOST entity:
    - ADD KNOWNHOST: adds a new known host to the database.
    - ALTER KNOWNHOST: changes parameters for an existing known host.
    - DELETE KNOWNHOST: deletes an existing known host.
    - FREEZE KNOWNHOST: freezes a known host, rendering it inactive.
    - INFO KNOWNHOST: shows information about a known host or a set of known hosts.
    - RENAME KEY: renames a known host.
    - THAW KNOWNHOST: thaws a user, making it active again.

These commands will be discussed in detail in the following subsections. Please also see "Database for Client Mode" in "The SSH User Database" chapter, for an overview of the database content.

ASSUME USER

The KEY, KNOWNHOST and PASSWORD entities are associated with a single Guardian system user. In the case of the KNOWNHOST entity, the reserved user name ALL is also allowed to specify that a KNOWNHOST can be accessed by all Guardian users.

The ASSUME user command sets a user name as default for the following commands. Subsequent commands that allow the specification of a user name can therefore be abbreviated. The command has the following syntax:

```
ASSUME USER [<system-user-name>]
```

If no user name is specified, the command will display the current value assumed.

Otherwise it will change the value to the user name provided.

The User ALL

The username ALL is reserved to specify all local NonStop system users in conjunction with the KNOWNHOST entity. If a KNOWNHOST is set to the user ALL, it means that all local system users can access that host. Note that the user ALL has no special meaning for the KEY or PASSWORD entity.

INFO SYSTEM-USER

KEY, KNOWNHOST and PASSWORD entities are each maintained via a set of CLIENT mode commands like GENERATE KEY, ALTER KNOWNHOST and FREEZE PASSWORD. The INFO SYSTEM-USER lists all KEY, KNOWNHOST and PASSWORD records assigned (owned) by a specific local Guardian system user. Both the KEY and the KNOWNHOST entity are associated with a single Guardian system user. Besides providing an overview of the system user related client mode records, the INFO SYSTEM-USER lists additionally the remote ssh user names (i.e. keys to the damon mode USER records) that are mapped to a specific local system user or that are configured with OWNER field set to the specific local system user.

The command has the following syntax:

```
INFO SYSTEM-USER [<system-user-name> | <partial-system-user-name>* | *]
                 [,DETAIL]
```

If no user name is specified, the command will display the entries for the current (or assumed) system

user.

The wildcard character '*' can be used alone to select all entries or it can be preceded by a name prefix to select all entries where the system user name starts with the given prefix.

The DETAIL attribute can be specified, if detailed information is needed.

The individual attributes have the following meaning:

<system-user-name>

A valid GUARDIAN user. If <system-user-name> is omitted, then either the user being set with a previously issued ASSUME USER command or the issuer of the INFO SYSTEM-USER command will be used as the default.

<partial-system-user-name>

A prefix that is used to match system users owning knownhost, password and key entries in the SSHCTL database.

Client Mode Commands Operating on the KEY Entity

ALTER KEY

The ALTER KEY command changes one or more attributes of an existing user private key and has the following syntax:

```
ALTER KEY  [<system-user-name>:]<key-name>
           [, COMMENT "<comment>"]
           [, LIVE-DATE <date-time>]
           [, EXPIRE-DATE <date-time>]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

This refers to a valid GUARDIAN user who owns the key in the SSH key store. If <system-user-name> is omitted, either the user being set in a previously issued ASSUME USER command or the issuer of the ALTER KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

<key-name>

This refers to the name of the key owned by the current user. The key name cannot be altered.

<date time>

Date or date and time in either of the following formats:

- DD Mon YYYY hh:mm
- "DDMonYY,hh:mm"
- DD Mon YYYY
- DDMonYY

The second format requires surrounding quotes because it contains a comma (commas are separators in SSHCOM).

COMMENT

This optional attribute is used to associate additional textual information with the key.

LIVE-DATE

This optional attribute is used to set the LIVE-DATE (not-valid-before date) for the key. This attribute can only be set if the life-cycle policy for User Private Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to FIXED, then field LIVE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to VARIABLE, then every user can change field LIVE-DATE for those keys the user owns.

EXPIRE-DATE

This optional attribute is used to set the EXPIRE-DATE (not-valid-after date) for the key. This attribute can only be set if the life-cycle policy for User Private Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to FIXED, then field EXPIRE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to VARIABLE, then every user can change field EXPIRE-DATE for those keys the user owns.

DELETE KEY

The DELETE KEY command deletes a key from the database and has the following syntax:

DELETE KEY [<system-user-name> :]<key-name>

The individual attributes have the following meaning and syntax:

<system-user-name>

This refers to a valid GUARDIAN user who owns the key in the SSH key store. If <system-user-name> is omitted, either the user being set in a previously issued ASSUME USER command or the issuer of the ALTER KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can delete keys from other users.

<key-name>

This refers to the name of the key to be deleted.

EXPORT KEY

The EXPORT KEY command exports a single private/public key pair or just the public key of a key pair into a GUARDIAN or OSS file. If both keys are exported (private and public), then they are stored into a single file.

The command has the following syntax:

```
EXPORT KEY [<system-user-name>:]<key-name>
  ,FILE {<GUARDIAN-file-name> | "<OSS-file-name>" | <OSS-file-name> }
           [, PASSPHRASE "<passphrase>"]
           [, FORMAT { OPENSSH | SSH2 }]
           [, PRIVATE]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

This refers to a valid GUARDIAN user who owns the key in the SSH key store. If <system-user-name> is omitted, either the user being set in a previously issued ASSUME USER command or the issuer of the ALTER KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

<key-name>

The name of the key owned by the current user.

FILE

The name of the GUARDIAN or OSS file that will hold the exported key. If the OSS file name contains spaces, it must be enclosed in double quotes.

PASSPHRASE

This attribute is relevant only if the PRIVATE attribute is set.

It configures the optional passphrase to secure the resulting private key file. The passphrase must be enclosed in double quotes (i.e. "..."). If the PASSPHRASE attribute is omitted, the private key can be retrieved by anyone who has read access to the file.

FORMAT

The format of the resulting key file. Format can be either OPENSSH or SSH2. If this attribute is omitted, SSH2 will be used as the default. Export of the private key part is not supported when exporting in format SSH2.

PRIVATE

If this attribute is specified, the full private key will be exported, otherwise only the public part of the key will be exported.

Note: Exporting a private key may result in a compromise of security. Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can export private keys.

FREEZE KEY

The FREEZE KEY command freezes a key. A local SFTP client cannot connect to a remote host using a key that has a status set as frozen. The key won't enable access until it is thawed using the THAW KEY command.

The command has the following syntax:

```
FREEZE KEY [<system-user-name>:]<key-name>
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the key entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the FREEZE KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the known host name that follows.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can freeze a key entry for other users.

<key-name>

The name of the key to be frozen.

GENERATE KEY

This command is used to generate a private/public key pair. The generated key is added to the SSH2 key store. The command has the following syntax:

```
GENERATE KEY [<system-user-name>:]<key-name>
              , TYPE {RSA | DSA}
             [, BITS <number>]
             [, COMMENT "<comment>"]
             [, LIVE-DATE <date-time>]
             [, EXPIRE-DATE <date-time>]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the key in the SSH key store. If <system-user-name> is omitted, either the user being set in a previously issued ASSUME USER command or the issuer of the ALTER KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

<key-name>

The name of the key owned by the current user.

<date time>

Date or date and time in either of the following formats:

- DD Mon YYYY hh:mm
- "DDMonYY,hh:mm"

- DD Mon YYYY
- DDMonYY

The second format requires surrounding quotes because it contains a comma (commas are separators in SSHCOM).

TYPE

Specifies the type of the key to be generated. Users can choose from RSA and DSA.

BITS

Optional attribute to set the key length. If this attribute is omitted, the generated key will have a default length of 1024 bits. Allowed values are 1024 and 2048 bits only.

COMMENT

This optional attribute is used to associate additional textual information with the generated key.

LIVE-DATE

This optional attribute is used to set the LIVE-DATE (not-valid-before date) for the key. This attribute can only be set if the life-cycle policy for User Private Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to FIXED, then field LIVE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to VARIABLE, then every user can change field LIVE-DATE for those keys the user owns.

EXPIRE-DATE

This optional attribute is used to set the EXPIRE-DATE (not-valid-after date) for the key. This attribute can only be set if the life-cycle policy for User Private Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to FIXED, then field EXPIRE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to VARIABLE, then every user can change field EXPIRE-DATE for those keys the user owns.

IMPORT KEY

This command imports a private/public key pair from a file into the SSH2 key store. It has the following syntax:

```
IMPORT KEY [<system-user-name>:]<key-name>
            , FILE <filename>
           [, PASSPHRASE "<passphrase>" ]
           [, COMMENT "<comment>"]
           [, LIVE-DATE <date-time>]
           [, EXPIRE-DATE <date-time>]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the key in the SSH key store. If <system-user-name> is omitted, either the user being set in a previously issued ASSUME USER command or the issuer of the ALTER KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

<key-name>

The name of the key owned by the current user. Multiple owners can have keys with same name.

FILE

The name of the file that holds the private key to be imported.

PASSPHRASE

The optional passphrase associated with the private key file. The passphrase must be enclosed in double quotes (i.e. "..."). If the PASSPHRASE attribute is not specified, it is assumed that the key file is accessible without a passphrase.

<date time>

Date or date and time in either of the following formats:

- DD Mon YYYY hh:mm
- "DDMonYY,hh:mm"
- DD Mon YYYY
- DDMonYY

The second format requires surrounding quotes because it contains a comma (commas are separators in SSHCOM).

COMMENT

This optional attribute is used to associate additional textual information with the imported key.

LIVE-DATE

This optional attribute is used to set the LIVE-DATE (not-valid-before date) for the key. This attribute can only be set if the life-cycle policy for User Private Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to FIXED, then field LIVE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to VARIABLE, then every user can change field LIVE-DATE for those keys the user owns.

EXPIRE-DATE

This optional attribute is used to set the EXPIRE-DATE (not-valid-after date) for the key. This attribute can only be set if the life-cycle policy for User Private Keys is enabled (determined by SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY). If SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to FIXED, then field EXPIRE-DATE can be modified by the SUPER.SUPER user only (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access. In case the SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY is set to VARIABLE, then every user can change field EXPIRE-DATE for those keys the user owns.

INFO KEY

This command provides information about a single key or a set of keys in the SSH2 key store. It has the following syntax:

```
INFO KEY [<system-user-name>:]{<key-name> | *} [, DETAIL]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the key in the SSH key store. If <system-user-name> is omitted, either the user being set in a previously issued ASSUME USER command or the issuer of the ALTER KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

<key-name>

The name of the key owned by the current user. A '*' as part of the key name will be interpreted as a wildcard character, and information about all key names matching the wildcard character will be displayed.

OUTPUT format of INFO KEY command

If used without the DETAIL modifier, INFO KEY will provide a brief summary for each key displayed. The following is an example of the output of INFO KEY:

```
% info key *:*
info key *:*

KEY                     TYPE USER              LIFE-CYCLE LAST-USE      STATUS
mytestkey                RSA mh                PENDING    *NONE*        THAWED
tst4                     RSA stus              PENDING    *NONE*        THAWED
new1                     RSA super.super       LIVE       08Jul11,18:22 THAWED
us2                      RSA super.super       EXPIRED    *NONE*        THAWED
tstky                    RSA tb                PENDING    *NONE*        THAWED
ky99                     RSA test              PENDING    *NONE*        THAWED
%
```

If used with the DETAIL modifier, INFO KEY will provide some detailed information about each key displayed. The following is an example of the output of INFO KEY, DETAIL:

```
% info key new1,detail
info key new1,detail

KEY                     TYPE USER              LIFE-CYCLE LAST-USE      STATUS
new1                     RSA super.super       LIVE       08Jul11,18:22 THAWED

  KEY new1
  COMMENT
  USER super.super
  TYPE RSA
  BITS 1024
  PUBLICKEY-FINGERPRINT
    MD5    e1:96:56:e2:d3:f1:96:3a:c6:00:78:6e:8f:4a:76:37
    BABBLE xicef-sineb-gopiv-byfeb-lahal-vidan-kimev-cekoh-zylyp-manav-zexix
  CREATION-DATE 04May11,22:40
  LIVE-DATE 01Jun11,00:00
  EXPIRE-DATE 31Aug11,12:30
  LIFE-CYCLE-STATE LIVE
  LAST-USE 08Jul11,18:22
  LAST-MODIFIED 08Jul11,19:01
  STATUS THAWED
%
```

The fields of the output of INFO KEY have the following meaning:

COMMENT

A comment as entered when generating, importing, or altering the key.

USER

The system user who owns the private key.

TYPE

The type of the key.

BITS

The key length in bits.

PUBLICKEY-FINGERPRINT

Both the MD5 and bubble-babble fingerprint of the public key.

CREATION-DATE

This attribute contains the creation date of a key and is automatically set when a key is generated or imported. If a key was generated or imported before the introduction of the CREATION-DATE attribute, the value will be shown as *NONE*, meaning ‘not set’.

LIVE-DATE

This optional attribute contains the date the key has gone or will go into state ‘LIFE’. The key is not valid before that date and will not be used for authentication. If a key was generated or imported before the introduction of the LIVE-DATE attribute, or if an attribute value was not specified in a GENERATE KEY or IMPORT KEY command, then the value will be shown as *NONE*, meaning ‘not set’. The field can be modified using the ALTER KEY command, depending on the value of SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY.

EXPIRE-DATE

This optional attribute contains the date the key has gone or will go into state ‘LIFE’. The key is not valid after that date and will no longer be used for authentication if the expiration date is reached. If a key was generated or imported before the introduction of the EXPIRE-DATE attribute, or if an attribute value was not specified in a GENERATE KEY or IMPORT KEY command, then the value will be shown as *NONE*, meaning ‘not set’. The field can be modified using the ALTER KEY command, depending on the value set of SSH2 parameter LIFECYCLEPOLICYPRIVATEUSERKEY.

LIFE-CYCLE-STATE

The value of field LIFE-CYCLE-STATE (the shortcut LIFE-CYCLE is used in the brief output of the INFO KEY command) is not actually held in the KEY database record but is determined from CREATION-DATE, LIVE-DATE and EXPIRE-DATE. The state ‘LIFE’ is assumed for keys generated or imported before the introduction of the user private key life-cycle.

LAST-USE

The timestamp of the last usage of the key.

LAST-MODIFIED

The timestamp of the last modification of the key.

STATUS

Whether the key is FROZEN or THAWED.

RENAME KEY

The RENAME KEY command is used to rename a key entry in the SSH database. A key entry can only be renamed by the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access or by the user who owns the key. The command has the following syntax:

```
RENAME KEY [<old-system-user-name>:]<old key name>,
           [<new-system-user-name>:]<new key name>
```

The individual attributes have the following meaning and syntax:

<old-system-user-name>

A valid GUARDIAN user who owns the key entry in the user database before renaming it. If <user name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the RENAME KEY command will be used as the default. If <user name> is specified, it MUST be followed by a ':' to separate it from the key name.

<old key name>

Specifies the name of a key entry, which must already exist in the user database, before it is renamed.

<new-system-user-name>

A valid GUARDIAN user who will own the key entry in the SSHCTL database after the rename. Only SUPER.SUPER users (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can issue a RENAME command where <new-system-user-name> is different from <old-system-user-name>.

If <old-system-user-name> and/or <new-system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the RENAME KEY command will be used as the default user.

If <old-system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

<new key name>

The new name of the key entry. A key entry with this name owned by the specified GUARDIAN user must NOT already exist in the user database.

THAW KEY

The THAW KEY command thaws a key. The command has the following syntax:

```
THAW KEY [<system-user-name>:]<key-name>
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the key entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the THAW KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name that follows.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can thaw a key entry for other users.

<key-name>

The name of the key to be thawed.

Client Mode Commands Operating on the PASSWORD Entity

ADD PASSWORD

The ADD PASSWORD command adds a new password to the database and has the following syntax:

```
ADD PASSWORD [<system-user-name>:]<remote-user>@<target-host>[:<target-port>]
             , { <word> | "<word> <word> ..." }]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid local GUARDIAN user who owns the password entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the ADD PASSWORD command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the known host name that follows.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can add a password entry for other users.

<remote-user>

The user name to be used on the remote system.

<target-host>

The DNS name or IP address of the target system.

<target-port>

The listening port of the remote SSH server. If this optional attribute is omitted, the default of 22 is used.

<word>

<word> is the password used to authenticate against the remote system. If the password contains spaces, it has to be enclosed in double quotes.

ALTER PASSWORD

The ALTER PASSWORD command changes the comment attribute of an existing password and has the following syntax:

```
ALTER PASSWORD [<system-user-name>:]<remote-user>@<target-host>[:<target-port>]
               , { <word> | "<word> <word> ..." }]
```

The individual attributes are identical as in the ADD PASSWORD command, please see that section for details.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can alter a password entry for other users.

DELETE PASSWORD

The DELETE PASSWORD command deletes a password from the database and has the following syntax:

```
DELETE PASSWORD [<system-user-name>:]<remote-user>@<target-host>[:<target-port>]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid local GUARDIAN user who owns the password entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the ADD PASSWORD command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the known host name that follows.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can delete a password entry for other users.

<remote-user>

The user name to be used on the remote system.

<target-host>

The DNS name or IP address of the target system.

<target-port>

The listening port of the remote SSH server. If this optional attribute is omitted, the default of 22 is used.

FREEZE PASSWORD

The FREEZE PASSWORD command freezes a password. A local SFTP client cannot connect to a remote host using this password until this password entry is thawed using the THAW PASSWORD command.

The command has the following syntax:

```
FREEZE PASSWORD [<system-user-name>:]<remote-user>@<target-host>[:<target-port>]
```

The individual attributes are identical as in the DELETE PASSWORD command, please see that section for details.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can freeze a password entry for other users.

INFO PASSWORD

This command provides information about a single password or a set of passwords in the SSH2 key store. It has the following syntax:

```
INFO PASSWORD [<system-user-name>:]<remote-user>@<target-host>[:<target-port>]
[,DETAIL]
```

The attributes used to specify the password have the same meaning as in the DELETE PASSWORD command, please see that section for details.

A '*' as part of the remote user name will be interpreted as a wildcard character, and information about all password names matching the wildcard character will be displayed.

OUTPUT Format of INFO PASSWORD Command

If used without the DETAIL modifier, INFO PASSWORD will provide a brief summary for each password displayed. The following is an example of the output of INFO PASSWORD:

```
%info password *

PASSWORD                                             USER             STATUS
comf.us@10.0.0.194:55022                             superulrich      THAWED
comf.us@10.0.0.196                                   superulrich      THAWED
comf.us@[fe80::a00:8eff:fe00:d14e]:55022             superulrich      THAWED
%
```

If used with the DETAIL modifier, INFO PASSWORD will provide some detailed information about each password displayed. The following is an example of the output of INFO PASSWORD, DETAIL:

```
% info password comf.us@[fe80::a00:8eff:fe00:d14e]:55022,detail
info password comf.us@[fe80::a00:8eff:fe00:d14e]:55022,detail

PASSWORD                                             USER              STATUS
comf.us@[fe80::a00:8eff:fe00:d14e]:55022             superulrich       THAWED

  USERID@HOST comf.us@[fe80::a00:8eff:fe00:d14e]:55022
  USER superulrich
  LAST-USE 20Apr12,20:05
  LAST-MODIFIED 20Apr12,19:11
  STATUS THAWED
%
```

Specifying a prefix followed by a wildcard is supported:

```
% info password superu*:u*,detail
info password superu*:u*,detail

PASSWORD                                             USER              STATUS
us@10.0.0.196                                        superulrich       THAWED

  USERID@HOST us@10.0.0.196
  USER superulrich
  LAST-USE 20Apr12,20:13
  LAST-MODIFIED 20Apr12,20:12
  STATUS THAWED
%
```

The fields of the output of INFO PASSWORD have the following meaning:

USER

The system user who owns the password.

LAST-USE

The timestamp of the last usage of the password.

LAST-MODIFIED

The timestamp of the last modification of the password.

STATUS

Whether the password is FROZEN or THAWED.

RENAME PASSWORD

The RENAME PASSWORD command is used to rename a password entry in the SSH database. A password entry can only be renamed by the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access or by the user who owns the password. The command has the following syntax:

```
RENAME PASSWORD [<oldusername>:]<oldremoteuser>@<oldtargethost>[:<oldtargetport>],
                [<newusername>:]<newremoteuser>@<newtargethost>[:<newtargetport>]
```

A password entry with the old password name, identified by the sequence

```
[<oldusername>:]<oldremoteuser>@<oldtargethost>[:<oldtargetport>]
```

must exist. The entry with the new password name, identified by

```
[<newusername>:]<newremoteuser>@<newtargethost>[:<newtargetport>]
```

must not exist.

The individual attributes have the following meaning and syntax:

<oldusername>

A valid GUARDIAN user who owns the password entry in the user database before renaming it. If <oldusername> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the RENAME PASSWORD command will be used as the default. If <oldusername> is specified, it MUST be followed by a ':' to separate it from the password name.

<oldremoteuser>

A user name of the targeted system.

<oldtargethost>

The IP address or the DNS name of the targeted system.

<oldtargetport>

The listening port of the remote SSH server. If this optional attribute is omitted, the default of 22 is used.

<newusername>

A valid GUARDIAN user who will own the password entry in the SSHCTL database after the rename. Only SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can issue a RENAME command where <newusername> is different from <oldusername>.

If <oldusername> and/or <newusername> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the RENAME PASSWORD command will be used as the default user.

If <newusername> is specified, it MUST be followed by a ':' to separate it from the password name.

<newremoteuser>

A user name of the targeted system.

<newtargethost>

The IP address or the DNS name of the targeted system.

<newtargetport>

The listening port of the remote SSH server. If this optional attribute is omitted, the default of 22 is used.

THAW PASSWORD

The THAW PASSWORD command thaws a password. The command has the following syntax:

```
THAW PASSWORD [<system-user-name>:]<remote-user>@<target-host>[:<target-port>]
```

The individual attributes are identical as in the DELETE PASSWORD command, please see that section for details.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can thaw a password entry for other system users.

Client Mode Commands Operating on the KNOWNHOST Entity

ADD KNOWNHOST

The ADD KNOWNHOST command adds a new known host to the database and has the following syntax:

```
ADD KNOWNHOST [<system-user-name>:]<knownhost-name>
              , ADDRESSES {<ip-or-dns> | ( <ip-or-dns> [,<ip-or-dns>,]...) }
              , PORT <portnr>
              , PUBLICKEY {FINGERPRINT <fingerprint> |
                           FILE <file name>}
              , ALGORITHM {SSH-DSS | SSH-RSA}
              [, COMMENT { <word> | "<word> <word> ..." }]
              [, FROZEN]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the known host entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the ADD KNOWNHOST command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the known host name that follows.

The user name ALL means that all users can access that known host.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can add a known host entry for other users.

<knownhost-name>

The name of the known host to be added.

ADDRESSES

Specifies an IP address, a DNS name or a comma separated list of IP addresses or DNS names enclosed in parentheses, which identify the target host, which the publickey associated with this knownhost entry is accepted from.

PORT

The target port number of the remote host associated with this known host entry.

PUBLICKEY

Either the MD5 fingerprint of the known host's public key or the name of a file that contains the remote host's public key. The fingerprint can either be specified in MD5 or bubble-babble format.

ALGORITHM

Specifies the key exchange algorithm to be used. Valid values are SSH-DSS and SSH-RSA.

COMMENT

An optional comment associated with the known host entry. The comment must be enclosed in double quotes if it contains spaces.

FROZEN

If the FROZEN attribute is set, the known host entry is added but frozen. A local SFTP client cannot connect to the remote host on the specified port until this known host entry is thawed using the THAW KNOWNHOST command.

ALTER KNOWNHOST

The ALTER KNOWNHOST command changes one or more attributes of an existing known host and has the following syntax:

```
ALTER KNOWNHOST [<system-user-name>:]<knownhost-name>
                [, ADDRESSES <ip_or_dns> [,<ip_or_dns>,]...]
                [, PORT <portnr>]
                [, PUBLICKEY {FINGERPRINT <fingerprint> |
                             FILE <file name>} ]
                [, ALGORITHM {SSH-DSS | SSH-RSA} ]
                [, COMMENT { <word> | "<word> <word> ..." }]
```

The individual attributes are identical as in the ADD KNOWNHOST command, please see that section for details.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can alter a known host entry for other users.

DELETE KNOWNHOST

The DELETE KNOWNHOST command deletes a known host from the database and has the following syntax:

```
DELETE KNOWHOST [<system-user-name>:]<knownhost-name>
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the known host entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the ADD KNOWNHOST command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the known host name that follows.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can delete a known host entry for other users.

<knownhost-name>

The name of the known host to be deleted.

FREEZE KNOWNHOST

The FREEZE KNOWNHOST command freezes a known host. A local SFTP client cannot connect to the remote host on the specified port until this known host entry is thawed using the THAW KNOWNHOST command.

The command has the following syntax:

```
FREEZE KNOWNHOST [<system-user-name>:]<knownhost-name>
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the known host entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the FREEZE KNOWNHOST command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the known host name that follows.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can freeze a known host entry for other users.

<knownhost-name>

The name of the known host to be frozen.

INFO KNOWNHOST

This command provides information about a single known host or a set of known hosts in the SSH2 key store. It has the following syntax:

```
INFO KNOWNHOST [<system-user-name>:]{<knownhost-name> | *} [, DETAIL]
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the known host in the SSH key store. If <system-user-name> is omitted, either the user being set in a previously issued ASSUME USER command or the issuer of the ALTER KEY command will be used as the default. If <system-user-name> is specified, it MUST be followed by a ':' to separate it from the known host name.

<knownhost-name>

The name of the known host owned by the current user. A '*' as part of the known host name will be interpreted as wildcard character and information about all known host names matching the wildcard character will be displayed.

OUTPUT Format of INFO KNOWNHOST Command

If used without the DETAIL modifier, INFO KNOWNHOST will provide a brief summary for each known host displayed. The following is an example of the output of INFO KNOWNHOST:

```
% info knownhost *:*
info knownhost *:*

KNOWNHOST                                            KNOWNBY          STATUS
10.0.0.11.22                                         super.super      THAWED
10.0.0.194.55022                                     superulrich      THAWED
10.0.0.196.22                                        superulrich      THAWED
fe80::a00:8eff:fe00:d14e.55022                       superulrich      THAWED
npns01ipv6.54022                                     superulrich      FROZEN
%
```

If used with the DETAIL modifier, INFO KNOWNHOST will provide some detailed information about each known host displayed. The following is an example of the output of INFO KNOWNHOST, DETAIL:

```
% info knownhost superulrich:npns01ipv6.54022,detail
info knownhost superulrich:npns01ipv6.54022,detail

KNOWNHOST                                            KNOWNBY          STATUS
npns01ipv6.54022                                     superulrich      FROZEN

  KNOWNHOST npns01ipv6.54022
  COMMENT automatically added by SSH2
  KNOWNBY superulrich
  ADDRESSES npns01ipv6
  PORT 54022
  ALGORITHM ssh-dss
  PUBLICKEY-FINGERPRINT
    MD5    87:33:4c:98:3e:a4:cd:0c:40:0b:51:d8:0d:6f:f2:fd
    BABBLE xibod-gogif-deret-sezip-bymek-decam-gonyt-ripoc-fygyr-pobet-kaxox
  LAST-USE *NONE*
  LAST-MODIFIED 23Apr12,10:32
  STATUS FROZEN
%
```

The fields of the output of INFO KNOWNHOST have the following meaning:

COMMENT

A comment as entered when adding or altering the known host.

KNOWNBY

The system user who is allowed to connect to the known host.

ADDRESSES

Specifies a comma separated list of IP addresses or DNS names that identify the target host, from which the public key associated with this known host entry is accepted.

PORT

The target port number of the remote host associated with this known host entry

ALGORITHM

The key exchange algorithm to be used. Valid values are SSH-DSS and SSH-RSA.

PUBLICKEY

The MD5 and/or bubble-babble fingerprint of the known host's public key.

COMMENT

An optional comment associated with the known host entry. The comment must be enclosed in double quotes if it contains spaces.

LAST-USE

The timestamp of the last usage of the known host.

LAST-MODIFIED

The timestamp of the last modification of the known host.

STATUS

Whether the known host is FROZEN or THAWED.

RENAME KNOWNHOST

The RENAME KNOWNHOST command is used to rename a knownhost entry in the SSH database. A knownhost entry can only be renamed by the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access or by the user who owns the knownhost. The command has the following syntax:

```
RENAME KNOWNHOST [<old-system-user-name>:]<old knownhost name>,
                 [<new-system-user-name>:]<new knownhost name>
```

The individual attributes have the following meaning and syntax:

<old-system-user-name>

A valid GUARDIAN user who owns the key entry in the user database before renaming it. If <old-system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the RENAME KNOWNHOST command will be used as the default. If <old-system-user-name> is specified, it MUST be followed by a ':' to separate it from the knownhost name.

<old knownhost name>

Specifies the name of a knownhost entry, which must already exist in the user database, before it is renamed.

<new-system-user-name>

A valid GUARDIAN user who will own the key entry in the SSHCTL database after the rename. Only SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can issue a RENAME command where <new-system-user-name> is different from <old-system-user-name>.

If <old-system-user-name> and/or <new-system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the RENAME KNOWNHOST command will be used as the default user.

If <new-system-user-name> is specified, it MUST be followed by a ':' to separate it from the key name.

<new knownhost name>

The new name of the knownhost entry. A knownhost entry with this name owned by the specified GUARDIAN user must NOT already exist in the user database.

THAW KNOWNHOST

The THAW KNOWNHOST command thaws a known host. The command has the following syntax:

```
THAW KNOWNHOST [<system-user-name>:]<knownhost-name>
```

The individual attributes have the following meaning and syntax:

<system-user-name>

A valid GUARDIAN user who owns the known host entry in the user database. If <system-user-name> is omitted, either the user being set with a previously issued ASSUME USER command or the issuer of the ADD KNOWNHOST command will be used as the default. If <system-user-name> is specified it MUST be followed by a ':' to separate it from the known host name that follows.

Only the SUPER.SUPER user (unless explicitly denied in OBJECTTYPE USER record) or those configured with full SSHCOM access can thaw a known host entry for another user..

<knownhost-name>

The name of the known host to be thawed.

Status Commands

The current parameter configuration of the SSH2 process can be viewed via commands INFO SSH2 and INFO DEFINE. The configuration of the SSHCTL database entities like USERs, KNOWNHOSTs, etc. can be listed via INFO USER, INFO KNOWNHOSTs, etc.. There are other entities in the SSH2 process that are of interest, especially the entities defined by the SSH protocol, namely sessions and channels. For displaying status data about the SSH2 process, sessions, and channels a set of STATUS commands exists in mode DAEMON:

- Status Commands:
  - STATUS SSH2: displays SSH2 process status information .
  - STATUS SESSION: displays SSH session information.
  - STATUS CHANNEL: displays SSH channel information.
  - STATUS OPENER: displays information about processes that have opened the SSH2 process.

STATUS SSH2

Status information about the SSH2 process will be displayed. The command has the following syntax:

```
STATUS SSH2    [,DETAIL]
               [,WIDTH <width>]
               [,RECURSIVE]
               [,LOG-ONLY]
               [,SELECT ( [<attr>] [, <attr>] ... ) ]
```

The individual command options have the following meaning and syntax:

DETAIL

If the DETAIL flag is set, detailed information is displayed.

WIDTH

The number <width> is the maximum number of characters per output line. If WIDTH is not specified the default value 80 is assumed. In order to avoid a new line when the terminal is configured with line wrapping on, the line will only be filled with one character less than the specified width.

RECURSIVE

This attribute controls if the sessions, channels and opener are displayed as well. A hierarchy is assumed with SSH2 at the top, sessions below and channels below sessions. Openers are displayed below SSH2 as well, when RECURSIVE is specified.

LOG-ONLY

Normally the output of the STATUS command will be displayed at the terminal the SSHCOM was started. With LOG-ONLY flag set, the output will be written to the log file, if logging to a file is enabled.

SELECT

The SELECT option allows defining a specific set of attributes that will be displayed instead of the default attribute set (there are two default sets, one for detailed output and one for non-detailed output). An attribute name specified for <attr> must be one of the names displayed in the detailed status output.

STATUS SESSION

Status information about the currently existing ssh sessions in the SSH2 process will be displayed. The command has the following syntax:

```
STATUS SESSION { <session-id> | *}
               [,DETAIL]
               [,WIDTH <width>]
               [,RECURSIVE]
               [,LOG-ONLY]
               [,SELECT ( [<attr>] [, <attr>] ... ) ]
               [,WHERE ( [<attr-filter>] [, <attr-filter>] ... ) ]
               [,FILTER-STATISTICS [ ONLY ]]
```

<session-id>

The internally assigned identifier (positive integer) of a session. Alternatively the wild card character '*' can be specified instead of a session id.

The individual options have the following meaning and syntax:

DETAIL

If the DETAIL flag is set, detailed information is displayed.

WIDTH

The number <width> is the maximum number of characters per output line. If WIDTH is not specified the default value 80 is assumed. In order to avoid a new line when the terminal is configured with line wrapping on, the line will only be filled with one character less than the specified width.

RECURSIVE

This attribute controls if the channels related to a specific session are displayed after each session. A hierarchy is assumed with SSH2 at the top, sessions below and channels below sessions. Openers are below SSH2.

LOG-ONLY

Normally the output of the STATUS command will be displayed at the terminal the SSHCOM was started. With LOG-ONLY flag set, the output will be written to the log file, if logging to a file is enabled.

SELECT

The SELECT option allows defining a specific set of attributes that will be displayed instead of the default attribute set (there are two default sets, one for detailed output and one for non-detailed output). An attribute name specified for <attr> must be one of the names displayed in the detailed status output.

WHERE

The WHERE option can be used to filter sessions. Only those sessions that fulfill all listed filter conditions <attr-filter> will be displayed. Each attribute filter must have the following format (the space characters surrounding the <operator> field are mandatory):

```
<attr> <operator> <value>
```

For information about <attr>, please see under option SELECT. The following operators are supported for <operator>: =, <> (for not equal), <, <=, > and >=

The value in <value> can be either a string, quoted string or number.

FILTER-STATISTICS

If it is of interest to determine the number of sessions matching the filter conditions, the option FILTER-STATISTICS can be specified. If the optional ONLY is added, then the status data is not displayed but just the total number of sessions and the number of matching sessions.

STATUS CHANNEL

Status information about the currently existing ssh channels in the SSH2 process will be displayed. The command has the following syntax:

```
STATUS CHANNEL { <channel-id> | *}
               [,DETAIL]
               [,WIDTH <width>]
               [,LOG-ONLY]
               [,SELECT ( [<attr>] [, <attr>] ... ) ]
               [,WHERE ( [<attr-filter>] [, <attr-filter>] ... ) ]
               [,FILTER-STATISTICS [ ONLY ]]
```

<channel-id>

The internally assigned identifier (positive integer) of a channel. Alternatively the wild card character '*' can be specified instead of a channel id.

The individual options have the following meaning and syntax:

DETAIL

If the DETAIL flag is set, detailed information is displayed.

WIDTH

The number <width> is the maximum number of characters per output line. If WIDTH is not specified the default value 80 is assumed. In order to avoid a new line when the terminal is configured with line wrapping on, the line will only be filled with one character less than the specified width.

LOG-ONLY

Normally the output of the STATUS command will be displayed at the terminal the SSHCOM was started. With LOG-ONLY flag set, the output will be written to the log file, if logging to a file is enabled.

SELECT

The SELECT option allows defining a specific set of attributes that will be displayed instead of the default attribute set (there are two default sets, one for detailed output and one for non-detailed output). An attribute name specified for <attr> must be one of the names displayed in the detailed status output.

WHERE

The WHERE option can be used to filter channels. Only those channels that fulfill all listed filter conditions <attr-filter> will be displayed. Each attribute filter must have the following format (the space characters surrounding the <operator> field are mandatory):

```
<attr> <operator> <value>
```

For information about <attr>, please see under option SELECT. The following operators are supported for <operator>: =, <> (for not equal), <, <=, > and >=

The value in <value> can be either a string, quoted string or number.

FILTER-STATISTICS

If it is of interest to determine the number of channels matching the filter conditions, the option FILTER-STATISTICS can be specified. If the optional ONLY is added, then the status data is not displayed but just the total number of channels and the number of matching channels.

STATUS OPENER

Status information about the currently existing openers, i.e. processes that have opened the SSH2 process will be displayed. The command has the following syntax:

```
STATUS OPENER { <opener-id> | *}
               [,DETAIL]
               [,WIDTH <width>]
               [,LOG-ONLY]
               [,SELECT ( [<attr>] [, <attr>] ... ) ]
               [,WHERE ( [<attr-filter>] [, <attr-filter>] ... ) ]
               [,FILTER-STATISTICS [ ONLY ]]
```

<opener-id>

The internally assigned identifier (positive integer) of an opener. Alternatively the wild card character '*' can be specified instead of an opener id.

The individual options have the following meaning and syntax:

DETAIL

If the DETAIL flag is set, detailed information is displayed.

WIDTH

The number <width> is the maximum number of characters per output line. If WIDTH is not specified the default value 80 is assumed. In order to avoid a new line when the terminal is configured with line wrapping on, the line will only be filled with one character less than the specified width.

LOG-ONLY

Normally the output of the STATUS command will be displayed at the terminal the SSHCOM was started. With LOG-ONLY flag set, the output will be written to the log file, if logging to a file is enabled.

SELECT

The SELECT option allows defining a specific set of attributes that will be displayed instead of the default attribute set (there are two default sets, one for detailed output and one for non-detailed output). An attribute name specified for <attr> must be one of the names displayed in the detailed status output.

WHERE

The WHERE option can be used to filter openers. Only those openers that fulfill all listed filter conditions <attr-filter> will be displayed. Each attribute filter must have the following format (the space characters surrounding the <operator> field are mandatory):

```
<attr> <operator> <value>
```

For information about <attr>, please see under option SELECT. The following operators are supported for <operator>: =, <> (for not equal), <, <=, > and >=

The value in <value> can be either a string, quoted string or number.

FILTER-STATISTICS

If it is of interest to determine the number of openers matching the filter conditions, the option FILTER-STATISTICS can be specified. If the optional ONLY is added, then the status data is not displayed but just the total number of openers and the number of matching openers.

Statistics Related Commands

Sometimes it is of interest to investigate the activity of ssh sessions in more detail, e.g. to view progress of file transfers. The progress feature can be enabled for each individual sftp session at the sftp prompt. With the introduction of the STATISTICS SESSION command the activity of all sessions handled by an SSH2 process can be displayed.

The commands ENABLE STATISTICS and DISABLE STATISTICS allow switching on and off the gathering of statistics data. Other commands are STATUS STATISTICS and RESET STATISTICS.

STATISTICS SESSION

The SSHCOM command has the following syntax:

```
{STATISTICS | STATS} SESSION { <session-id> | *}
                             [,DETAIL]
                             [,WIDTH <width>]
                             [,LOG-ONLY]
```

<session-id>

The internally assigned identifier (positive integer) of a session. Alternatively the wild card character '*' can be specified instead of a session id.

The individual options have the following meaning and syntax:

DETAIL

If the DETAIL flag is set, detailed information is displayed.

WIDTH

The number <width> is the maximum number of characters per output line. If WIDTH is not specified the default value 80 is assumed. In order to avoid a new line when the terminal is configured with line wrapping on, the line will only be filled with one character less than the specified width.

LOG-ONLY

Normally the output of the STATS command will be displayed at the terminal the SSHCOM was started. With LOG-ONLY flag set, the output will be written to the log file, if logging to a file is enabled.

DISABLE STATISTICS

Disables gathering of statistics data.

Syntax:

```
DISABLE {STATISTICS | STATS}
```

ENABLE STATISTICS

Enables gathering of statistics data.

Syntax:

```
ENABLE {STATISTICS | STATS}
```

RESET STATISTICS

Resets statistics counters/rates.

Syntax:

```
RESET {STATISTICS | STATS}
```

STATUS STATISTICS

Displays status of statistics, e.g. if gathering statistics is enabled. If the DETAIL flag is set, detailed information is displayed.

The SSHCOM command has the following syntax:

```
STATUS {STATISTICS | STATS} [,DETAIL]
```

Abort Session Command

In rare cases it may be required for an administrator to stop a session, e.g. because a user process was started in the wrong CPU or is using too much CPU or causing an unexpected high data throughput. Stopping a session can be achieved via the ABORT SESSION command.

The Syntax for the ABORT SESSION command is as follows:

```
ABORT SESSION <session-id>
```

<session-id>

The internally assigned identifier (positive integer) of a session. Wild card character '*' cannot be specified instead of a session id.

Only users with full SSHCOM access are allowed to execute the ABORT SESSION command.

Warning: Any unsaved changes made by processes related to the aborted session may be lost.

SSH and SFTP Client Reference

Introduction

The SSH2 package provides an SSH and SFTP client program to interact with SSH daemons on other systems. The clients programs will communicate with the SSH2 process, which will create the actual SSH session to the remote daemon. This chapter describes the usage of the SSH and SFTP client and assumes an SSH2 process is already running.

Starting the Guardian Client Programs

The clients for Guardian have the following filenames:

- SSH
- SFTP

The programs are simply started from TACL using the RUN command.

A typical command to establish an SSH session with a remote SSH daemon will look as follows:

```
$MH SSH 23> RUN ssh comf.mh@10.0.0.198 ls -l
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
Server did not accept any of your private keys in the key store.
Trying password authentication.
Enter comf.mh@10.0.0.198's password:
Add password for comf.mh@10.0.0.198 to the password store (yes/no)? no
total 955646
-rw-r--r--   1 COMF.MH          COMF        1000 Jan 18 11:28 a1000
-rw-r--r--   1 COMF.MH          COMF       10000 Sep 22  2004 a10000
-rw-r--r--   1 COMF.MH          COMF     1000000 Sep 22  2004 a1000000
...
$MH SSH 24>
```

Example with IPv6 address:

```
$DATA1 TEST 23> > run ssh comf.us@fe80::a00:8eff:fe00:d14e ls -l /G/us/temp
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
GSSAPI authentication disabled.
You have no private keys in the key store.
Trying password authentication.
Enter comf.us@fe80::a00:8eff:fe00:d14e's password:
Add password for comf.us@[fe80::a00:8eff:fe00:d14e]:54022 to the password store
(yes/no)? no
total 21933
-rwxr-xr-x    1 SUPER.SUPER        SUPER      38662 Apr 16 14:22 abc
-rwxr-xr-x    1 SUPER.SUPER        SUPER       2222 Nov 23  2010 c
-rwxr-xr-x    1 SUPER.SUPER        SUPER    11183778 Jan 20 09:24 crypto
-rwxr-xr-x    1 SUPER.SUPER        SUPER       2286 Sep 30  2011 test
-rwxr-xr-x    1 SUPER.SUPER        SUPER       2284 Sep 30  2011 test1
$DATA1 TEST 24>
```

A typical command to establish an SFTP session with a remote SSH daemon will look as follows:

```
$DATA1 MHSSH 20> run sftp m.horst@10.0.0.201
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
Connecting to 10.0.0.201...
You have no private keys in the key store.
Trying password authentication.
Enter m.horst@10.0.0.201's password:
Add password for m.horst@10.0.0.201 to the password store (yes/no)? no
sftp>
```

Example using IPv6 address:

```
> run sftp comf.us@~[fe80::a00:8eff:fe00:d14e~]
SFTP client version T9999H06_22Jan2014_comForte_SFTP_0097
Connecting to fe80::a00:8eff:fe00:d14e via SSH2 process $SSH00 ...
GSSAPI authentication disabled.
You have no private keys in the key store.
Trying password authentication.
Enter comf.us@fe80::a00:8eff:fe00:d14e's password:
Add password for comf.us@[fe80::a00:8eff:fe00:d14e]:54022 to the password store
(yes/no)? no
sftp>
```

The tilde characters are required if #INFORMAT is set to TACL; otherwise the square brackets must be used without tilde.

Starting the OSS Client Programs

The OSS object files of the SSH and SFTP client programs are delivered together with the other SSH implementation files. Therefore, the object files will initially be placed on the SSH2 installation subvolume. The clients for OSS have the following filenames:

- SSHOSS
- SFTPOSS

To start a client under OSS, there are a few choices:

- Start the program by specifying the full path on the shell, i.e.

```
>/G/system/comfssh/sshoss
>/G/system/comfssh/sftposs
```

- Create a symbolic link to the OSS program file in a directory which is included in the default search path under OSS, e.g.

```
>ln –s /G/system/comfssh/sshoss /usr/bin/ssh
>ln –s /G/system/comfssh/sftposs /usr/bin/sftp
```

- Copy the program file to a directory which is included in the default search path under OSS
- Copy the program file to a location of your choice and add that location to the default search path

In the subsequent sections of this chapter, we will assume the client program files are part of your current search path under the OSS shell.

If you start the program without any parameters, it will display a brief syntax summary and terminate:

```
> sshoss
Usage: sshoss [options] host [command]
Options:
  -l user     Log in using this user name.
  -t          Tty; allocate a tty even if command is given.
  -T          Do not allocate a tty.
  -V          Display version number only.
  -Z          Suppress ssh client banner.
```

```
  -q          Quiet; don't display any warning messages.
  -H string   Set prefix used for error messages. Default: no prefix.
  -J string   Set prefix used for info/warning messages. Default: no prefix.
  -K string   Set prefix used for prompt/query messages. Default: no prefix.  -c
cipher   Select encryption algorithm
  -m macs     Specify MAC algorithms for protocol version 2.
  -p port     Connect to this port.  Server must be on the same port.
  -L listen-port:host:port   Forward local port to remote address
  -R listen-port:host:port   Forward remote port to local address
              These cause sshoss to listen for connections on a port, and
              forward them to the other side by connecting to host:port.
              forward them to the other side by connecting to host:port.
  -C          Enable compression.
  -N          Do not execute a shell or command.
  -g          Allow remote hosts to connect to forwarded ports.
  -o 'option' Process the option as if it was read from a configuration file.
  -s          Invoke command (mandatory) as SSH2 subsystem.
  -S process  connect using this SSH2 process.
> sftposs
usage: sftposs [-vCZ] [-b batchfile] [-o ssh2_option]
            [-H error_prefix] [-J info_prefix] [-K query_prefix]
            [-B buffer_size] [-R num_requests] [-S ssh2 process]
            [user@]host[:file [file]]
>
```

Typical start of an SSH session from OSS to a remote system:

```
/tmp: sshoss u.sauer@linuxdevipv6
SSH client version T9999H06_22Jan2014_comForte_SSHOSS_0097
GSSAPI authentication disabled.
You have no private keys in the key store.
Trying password authentication.
Enter u.sauer@linuxdevipv6's password:
Add password for u.sauer@linuxdevipv6 to the password store (yes/no)? no
Linux linux-dev 2.6.32-40-server #87-Ubuntu SMP Tue Mar 6 02:10:02 UTC 2012 x86_64
GNU/Linux
Ubuntu 10.04.4 LTS

Welcome to the Ubuntu Server!

Last login: Sat Apr 21 11:28:48 2012 from 10.0.0.194
~u.sauer@linux-dev:~$
```

Example for initiating an SSH session from OSS to a remote NonStop server using an IPv6 address:

```
/home/test: sshoss -S '$SSH55' -oPort=54022 comf.us@fe80::a00:8eff:fe00:d14e
SSH client version T9999H06_22Jan2014_comForte_SSHOSS_0097
GSSAPI authentication disabled.
You have no private keys in the key store.
Trying password authentication.
Enter comf.us@fe80::a00:8eff:fe00:d14e's password:
Add password for comf.us@[fe80::a00:8eff:fe00:d14e]:54022 to the password store
(yes/no)? no

STN00 Connected to STN version B17  2012/04/23 12:36 \NPNS01.$PTY54.#ZWN0015
STN46 Secure SSH session: xterm password aes256-cbc hmac-sha1
STN81 Client IP address: fe80::a00:8eff:fe00:d14e port 4196
STN82 SSH external user comf.us, Guardian system user COMF.US

STN44 Application has connected to this window

/G/DATA1/USHOME:
```

Example for starting SFTPOSS client using IPv6 address:

```
sftposs u.sauer@[fe80::250:56ff:fea7:4bdc]
SFTPOSS client version T9999H06_22Jan2014_comForte_SFTPOSS_0097
Connecting to fe80::a00:8eff:fe00:d14e via SSH2 process $SSH01 ...
GSSAPI authentication disabled.
You have no private keys in the key store.
Trying password authentication.
Enter comf.us@fe80::a00:8eff:fe00:d14e's password:
Add password for comf.us@[fe80::a00:8eff:fe00:d14e]:54022 to the password store
(yes/no)? no
sftp>
```

Configuring the SSH2 Process to Use

As mentioned earlier, the SSH and SFTP clients will interact with a running instance of the SSH2 object file. There are multiple ways to specify which instance to use:

- The –S runtime option will explicitly choose a specific instance by its process name. The following example starts an SFTP client picking the SSH2 instance with the process name $SSH1 (please note that under OSS the process name is embedded into single quotes to allow the special character $ to be used as part of a shell command):

```
>: sftposs -S '$ssh1' burgt@10.0.0.201
Connecting to 10.0.0.201...
sftp>
```

- By setting an environment variable named SSH2PREFIX in the client environment you can activate a heuristic to pick an SSH2 process depending on the CPU number it is running in. Please refer to "Load-Balancing Outbound SSH Sessions" in the chapter "Configuring and Running SSH2" for details.
- By setting an environment variable SSH2_PROCESS_NAME in the OSS shell specifying the SSH2 process the client should use.
- By adding a define =SSH2^PROCESS^NAME, CLASS MAP and the SSH2 process name set as FILE value.

Inquiring User Name If Not Supplied

The SSH[OSS] and SFTP[OSS] clients accept argument user@host as well as just host. If no user is specified the current user, i.e. the user who started the client, is taken as default value. This default can be changed via environment variable INQUIREUSERNAMEIFNOTSUPPLIED, which must be defined in the environment (TACL/shell) the clients are started from.

If PARAM/environment variable INQUIREUSERNAMEIFNOTSUPPLIED is set to true and the username was not specified, the SFTP[OSS] and SSH[OSS] clients now prompt the user for the username:

```
> ssh 10.0.0.196
comForte SSH client version T9999G06_22Jan2014_SSH_0097
User name @10.0.0.196: test
You have no private keys in the key store.
Trying password authentication.
Enter test@10.0.0.196's password:
...
```

If the user just hits return the default user name applies. If the PARAM/environment variable INQUIREUSERNAMEIFNOTSUPPLIED is not defined or is set to value FALSE the default user name is assumed as well (i.e. the behavior is then identical before introduction of INQUIREUSERNAMEIFNOTSUPPLIED).

Suppressing the Banner printed by Clients

When SSH[OSS] and SFTP[OSS] clients print a banner containing the version and name of the ssh client, e.g. like:

```
comForte SSH client version T9999G06_22Jan2014_SSH_0097
```

This banner can be suppressed by setting Boolean parameter SUPPRESSCLIENTBANNER in the client environment, i.e. via PARAM in a TACL environment:

```
PARAM SUPPRESSCLIENTBANNER TRUE
```

and via environment variable in OSH environment:

```
export SUPPRESSCLIENTBANNER=TRUE
```

Automating the SFTP/SSH clients

SSH[OSS] and SFTP[OSS] clients are normally used directly by humans but sometimes it is required to automate the control of these clients, e.g. by setting IN and OUT of a client to a controlling program or script. In this case it is helpful to differentiate between messages printed by the client during startup/connection phase and other data. The following new parameters must be set in the client environment (PARAM under TACL or environment variable under OSH).

Parameter	Meaning
SSHERRORPREFIX	String that is printed as prefix for an error message
SSHINFOPREFIX	String that is printed as prefix for informational messages
SSHQUERYPREFIX	String that is printed as prefix for queries (prompts)

For each of these parameters a corresponding option is supported by the clients as shown below:

Option	Meaning
-H "<errorprefix>"	String that is printed as prefix for an error message
-J "<infoprefix>"	String that is printed as prefix for informational messages
-K "<queryprefix>"	String that is printed as prefix for queries (prompts)

FILE I/O Parameters for SFTP/SFTPOSS

File operations executed on local disks can be influenced by setting specific parameters in the environment of SFTP and SFTPOSS clients.

Currently the parameters set for the SSH2 process are not propagated to the SFTP/SFTPOSS clients, i.e. without setting the parameters in the client environment the default values for these parameters are used. Guardian file attributes can be exchanged between sftp client and sftp server. But other settings must be configured independently on both the client and the server side. This must happen in a non-conflicting way. For example: If client and server are using different delimiters to indicate the end of a record (relevant for edit files and structured files), then the result of a file transfer will not be as expected.

For details on these parameters, please see description in section "SSH2 Parameter Reference" in chapter "Configuring and Running SSH2"). The following table shows which parameter can be used in the client environment when sending or receiving files.

Parameter	Used when Sending	Used when Receiving	Dependency on SFTP Server
RECORDDELIMITER	Yes	Yes	Yes. The SFTP client prompt command ASCII can be used to achieve the same configuration.
SFTPEDITLINEMODE	No	Yes	No. Only relevant when files are written locally
SFTPEDITLINENUMBERDECIMALINCR	No	Yes	No. Only relevant when files are written locally
SFTPEDITLINESTARTDECIMALINCR	No	Yes	No. Only relevant when files are written locally
SFTPENHANCEDERRORREPORTING	Yes	Yes	Details about remote NonStop SFTP server depend on SFTPENHANCEDERRORREPORTING setting for SSH2 on remote NonStop system.
SFTPEXCLUSIONMODEREAD	Yes	No	No. Only relevant when files are read locally
SFTPMAXEXTENTS	No	Yes	No. Only relevant when files are written locally
SFTPPRIMARYEXTENTSIZE	No	Yes	No. Only relevant when files are written locally
SFTPSECONDARYEXTENTSIZE	No	Yes	No. Only relevant when files are written locally
SFTPUPSHIFTGUARDIANFILENAMES	No	Yes	No. Only relevant when files are written locally

SSH Client Command Reference

Note: The SSH protocol is a complex protocol with many features. This Reference Manual only provides an overview about some features, for detailed information beyond this manual please refer to publications such as SSH, the Secure Shell, 2nd Edition by Daniel J. Barrett; Robert G. Byrnes; Richard E. Silverman (O'Reilly).

The SSH[OSS] Client is used for the following purposes:

- Start a SSH shell to control a remote system. A shell is an encrypted communication channel between two untrusted hosts over an insecure network which allows the client to control the server – similar to TACL/TELNET in the NonStop™ environment.
- Execute a command on the remote system.
- Start a port forwarding daemon process. Port forwarding is a way to “tunnel” unencrypted protocols over an SSH session so that they become encrypted.

Command-Line Reference

The SSH client allows you to specify some parameters on the command line. Starting the client without any parameters provides a syntax summary:

```
$US SSH89 4> run ssh
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
Usage: ssh [options] [user@]host [command]
Options:
  -l user     Log in using this user name.
  -t          Tty; allocate a tty even if command is given.
  -T          Do not allocate a tty.
  -V          Display version number only.
  -Z          Suppress ssh client banner.
  -q          Quiet; don't display any warning messages.
  -H string   Set prefix used for error messages. Default: no prefix.
  -J string   Set prefix used for info/warning messages. Default: no prefix.
  -K string   Set prefix used for prompt/query messages. Default: no prefix.
  -c ciphers  Select encryption algorithms
  -m macs     Specify MAC algorithms
```

```
  -p port     Connect to this port. Server must be on the same port.
  -L listen-port:host:port   Forward local port to remote address
  -R listen-port:host:port   Forward remote port to local address
              These cause ssh to listen for connections on a port, and
              forward them to the other side by connecting to host:port.
  -C          Enable compression.
  -N          Do not execute a shell or command.
  -g          Allow remote hosts to connect to forwarded ports.
  -o 'option' Process the option as if it was read from a configuration file.
  -s          Invoke command (mandatory) as SSH2 subsystem.
  -S process  connect using this SSH2 process.
STOPPED: $Z3PT
CPU time: 0:00:00.007
2: Process terminated with fatal errors or diagnostics
```

Termination Info: 1$US SSH89 5>

General Runtime options

-l user

Specify the user to log in as on the remote machine.

-V

Display version number only, then terminate.

-Z

The banner normally printed by the ssh client is suppressed (line "SSH client version T9999H06_22Jan2014_comForte_SSH_0097" in the above example). The suppression of the client banner can also be achieved by specifying a PARAM/environment variable SUPPRESSCLIENTBANNER with possible values 0 for false and 1 for true (the -Z option takes precedence over the PARAM/environment variable).

-q

Quiet mode: No warning or error messages are printed.

-c ciphers

Specify a comma-separated list of ciphers for encrypting the session. Currently the following ciphers are supported:

- aes256-cbc: AES (Rijndael) in CBC mode, with 256-bit key
- aes128-cbc: AES with 128-bit key
- twofish256-cbc: Twofish in CBC mode, with 256-bit key
- twofish128-cbc: Twofish with 128-bit key
- twofish-cbc: alias for "twofish256-cbc" (Note: this is being retained for historical reasons)
- blowfish-cbc: Blowfish in CBC mode
- 3des-cbc: three-key 3DES in CBC mode
- arcfour: the ARCFOUR stream cipher
- cast128-cbc: CAST-128 in CBC mode

If this option is not specified, the client will negotiate a cipher from list configured for the SSH2 server using the CIPHERS parameter.

-m macs

Specify a comma-separated list of message authentication algorithm for the session. Currently the following MACs are supported:

- hmac-sha1: HMAC-SHA1 (digest length=key length=20 bytes=160 bits)
- hmac-md5: HMAC-MD5 (digest length=key length=16 bytes=128 bits)
- hmac-sha1-96: first 96 bits of HMAC-SHA1 (digest length=12 bytes=96 bits, key length=20 bytes=160 bits)
- hmac-md5-96: first 96 bits of HMAC-MD5 (digest length=12 bytes=96 bits, key length=16 bytes=128 bits)

If this option is not specified, the client will negotiate a cipher from list configured for the SSH2 server using the MACS parameter.

-p port

The port to connect to on the remote host.

-C

Requests compression of all data (including stdin, stdout, stderr, and data for forwarded connections). The compression algorithm is the same used by gzip. Compression is desirable on slow connections, but will only slow down things on fast networks.

-o *option*

Set a configuration option for the SSH client

The following options are supported:

- BINDADDRESS=*address*

  The local address used for outgoing connections. Useful if the SSH2 process is configured with the unspecified address (0.0.0.0 or 0::0) for parameter INTERFACEOUT or multiple IP addresses are configured in INTERFACEOUT, the TCP/IP process is configured with more than one subnet and a specific local address needs to be used (e.g. due to firewall configuration restrictions).

- IDENTITY=*keyname*

  Use this option to select a specific KEY for authentication to the remote system. By default all KEYs that you have generated using the SSHCOM GENERATE KEY command will be presented to the remote host for publickey authentication. However, some servers will deny authentication after a maximum number of inacceptable keys are presented, which can create a problem if you have many keys. To overcome this problem, use the IDENTITY option to present only the key that has been advertised as authorized key to the target server.

- PORT=*port*

  The port to connect to on the remote host. This option has the same effect as the –p command line option.

- COMPRESSION=TRUE|FALSE

  Specify weather data compression should be enabled on the SSH session. This option has the same effect as the –C command line option.

- CIPHERS=*ciphers*

  Specify a comma-separated list of ciphers for encrypting the session. This option has the same effect as the –c command line option.

- MACS=*macs*

  Specify a comma-separated list of MAC algorithms. This option has the same effect as the –m command line option.

- USER=*user*

  Specify the user to log in as on the remote machine. This option has the same effect as the –l command line option or the user runtime parameter.

- AllowedAuthentications=*methods*

Specify the authentication methods that are allowed for user authentication. The value is a comma separated list of method names (without any spaces). See SSH2 parameter CLIENTALLOWEDAUTHENTICATIONS for the possibility to restrict the ssh clients' authentication methods.

-S process

Connect using a specific SSH2 process. See section "Configuring the SSH2 Process to Use" for further details.

Runtime options relevant only when creating a shell

-t

Force pseudo-tty allocation. This can be used to execute arbitrary screen-based programs on a remote machine.

-T

Do not allocate a tty.

-s

Use this option to request invocation of a subsystem on the remote system. Subsystems are a feature of the SSH2 protocol which facilitate the use of SSH as a secure transport for other applications (e.g. sftp). The subsystem is specified as the remote command.

Runtime options relevant only for port forwarding

-L [ftp/]*listen-port:host:port*

Specifies that the given listen-port on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to listen-port on the local side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to host and port from the remote machine.

Specifying the ftp/ prefix will enable dynamic port forwarding of FTP sessions, forwarding both FTP control and data connections over the SSH session.

The –g (gateway) option controls weather all connections or only those originating from “localhost” will be forwarded.

-R [ftp/]*listen-port:host:port*

Specifies that the given listen-port on the remote (daemon) host is to be forwarded to the given host and port on the local side. This works by allocating a socket to listen to listen-port on the remote side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to host and port from the local machine.

Specifying the ftp/ prefix will enable dynamic port forwarding of FTP sessions, forwarding both FTP control and data connections over the SSH session.

The –g (gateway) option controls weather all connections or only those originating from “localhost” will be forwarded.

-N

Do not execute a shell or command. This is useful for just forwarding ports.

-g

Allows remote hosts to connect to local forwarded ports. By default, only connections originating from "localhost" (127.0.0.1) will be forwarded. Using –g will forward any connection.

Runtime options relevant only when automating SSH client

-H string

Set specific string used as prefix for error messages displayed by the SSH client during the connection phase. Double quotes can be used to define strings containing a space or special characters. The prefix for errors can also be specified via PARAM/environment variable SSHERRORPREFIX (the -H option takes precedence over the PARAM/environment variable). There is no specific error prefix defined as default.

-J string

Set specific string used as prefix for informational or warning messages displayed by the SSH client during the connection phase. Double quotes can be used to define strings containing a space or special characters. The prefix for infos/warnings can also be specified via PARAM/ environment variable SSHINFOPREFIX (the -J option takes precedence over the PARAM/environment variable). There is no specific info/warning prefix defined as default.

-K string

Set specific string used as prefix for prompt/query messages displayed by the SSH client during the connection phase. Double quotes can be used to define strings containing a space or special characters. The prefix for infos/warnings can also be specified via PARAM/environment variable SSHQUERYPREFIX (the -K option takes precedence over the PARAM/environment variable). There is no specific query prefix defined as default.

Using the SSH client to create a shell controlling a remote system

Creating a full shell

The following example shows how to connect to a Linux system and execute some commands on that system using the SSH client from Guardian:

```
$TB TBSSH79 7> run ssh -S $TBS79 burgt@10.0.0.12
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
You have no private keys in the key store.
Trying password authentication.
Enter burgt@10.0.0.12's password:
Add password for burgt@10.0.0.12 to the password store (yes/no)? no
Last login: Thu Jun  5 07:45:45 2008 from 10.0.3.98
Have a lot of fun...
burgt@np-dev02:~> pwd
/home/burgt
burgt@np-dev02:~> ls
abc        etestftp               etestsftp_old       glubwrap      t4gig_file
bin
burgt@np-dev02:~> exit
logout
$TB TBSSH79 8>
```

Note that for the first connection a KNOWNHOST will have to be configured for the remote system in able to connect. Also note that the password of the remote system was queried once and not stored in the database. The last command “exit” tells the remote system to end the shell session.

Executing a single command

The following example shows how to connect to a Linux system and execute a single command on that system using the SSH client from OSS:

```
$TB TBSSH79 8> run ssh -S $TBS79 burgt@10.0.0.12 pwd
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
You have no private keys in the key store.
Trying password authentication.
Enter burgt@10.0.0.12's password:
Add password for burgt@10.0.0.12 to the password store (yes/no)? yes
/home/burgt
$TB TBSSH79 9> run ssh -S $TBS79 burgt@10.0.0.12 pwd
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
/home/burgt
```

```
$TB TBSSH79 10>
```

Note that the password for the remote system is stored after the first issuing of the command and that the next time entering the password is no longer needed.

Using the SSH client to create a port forwarding daemon

The following example shows how to use port forwarding to tunnel a Telnet session between two NonStop systems through SSH to encrypt the network traffic. It is based on the following assumptions:

- An SSH2 daemon is installed on the remote NonStop system with Port forwarding allowed. That requires the parameter ALLOWTCPFORWARDING to be set to true.
- The IP address on the remote NonStop system is 10.0.0.198. A TELSERV is running on port 23 on that IP stack
- A guardian user named COMF.TB exists on the remote system

The concept of port forwarding can be applied to any TCP protocol which uses a single port on the server side of the connection.

Starting port forwarding on the client system

The following command will start a port forwarding daemon on the client system

```
$TB TBSSH79 13> run ssh -S $TBS79 -N -L 2323:127.0.0.1:23 comf.tb@10.0.0.198
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
You have no private keys in the key store.
Trying password authentication.
Enter comf.tb@10.0.0.198's password:
```

The client will not be active before the password is given at the prompt. The port forwarding client listens for incoming connections on port 2323. 127.0.0.1:23 is the IP address/port of TELSERV on the remote system from the perspective of the remote NonStop host.

Connecting to the port forwarding client with a Telnet client

The following command will direct local Telnet traffic to the port forwarding client who in turn will forward it to the remote NonStop system:

```
$TB TBSSH79 2> telnet 127.0.0.1 2323
TELNET Client - T9558H01 - (19MAR12) - (IPMAAH)
Copyright Tandem Computers Incorporated 2004
Trying...Connected to 127.0.0.1.
Escape character is '^]'.


WELCOME TO NPS762A [PORT $ZTC1 #23 WINDOW $ZTN1.#PTYKFEK]
TELSERV - T9553G06 - (24FEB2006) - (IPMAEF)

Available Services:

OSS      TACL      EXIT
Enter Choice>
```

The following log message will show up in the SSH2 log file indicating that the session was indeed forwarded over the SSH session:

```
$TBS79|08Jul08 07:54:46.08|50|\NPNS01.$Z0D3: forwarding TCP connection from
127.0.0.1:5030 to 127.0.0.1:23
```

Using the SSH client to create an FTP port forwarding daemon

To tunnel FTP connections through a SSH connection, the SSH implementation must apply additional logic to ensure that the data port is also encrypted. The following example shows the encryption of an FTP connection between two NonStop systems by tunneling it over an SSH session.

The example is based on the following assumptions:

- An SSH2 daemon is installed on the remote NonStop system with Port forwarding allowed. That requires the parameter ALLOWTCPFORWARDING to be set to true.
- The IP address on the remote NonStop system is 10.0.0.198. FTPSERV is configured through PORTCONF to take connections coming in on port 21 on that IP stack
- A guardian user named COMF.TB exists on the remote system

Starting FTP port forwarding on the client system

The following command will start a FTP port forwarding daemon on the client system

```
$TB TBSSH79 16> run ssh -S $TBS79 -N -L ftp/2121:127.0.0.1:21 comf.tb@10.0.0.198
SSH client version T9999H06_22Jan2014_comForte_SSH_0097
You have no private keys in the key store.
Trying password authentication.
Enter comf.tb@10.0.0.198's password:
```

The client will not be active before the password is given at the prompt. The port forwarding client listens for incoming connections on port 2121. 127.0.0.1:21 is the IP address/port of FTPSERV on the remote system from the perspective of the remote NonStop host. The “ftp/” string after the –L tells the SSH client to use additional FTP forwarding logic.

Connecting to the port forwarding client with a FTP client

The following command sequence will direct local FTP traffic to the port forwarding daemon and in effect create an encrypted FTP session between the two systems:

```
$TB TBSSH79 2> ftp 127.0.0.1 2121
FTP Client - T9552J01 - (30MAR2012) - COPYRIGHT TANDEM COMPUTERS INCORPORATED 2012
Connecting to 127.0.0.1.........Established.
220 NPS762A FTP SERVER T9552G07 (Version 3.x TANDEM 30NOV2005) ready.
Name (127.0.0.1:user): comf.tb
331 Password required for COMF.TB.
Password:
230 User COMF.TB logged in.  OSS API enabled
ftp> dir
200 command successful
150 Opening data connection for /bin/ls (127.0.0.1,4519d) (0 bytes).
total 9662
drwxrwxrwx   1 COMF.TB         COMF        4096 Jun 25 13:08 .
drwxrwxr-x   1 SUPER.SUPER     SUPER       4096 Jul 03 20:43 ..
-rw-------   1 COMF.TB         COMF        5430 May 08 16:40 .bash_history
-rw-rw-rw-   1 COMF.TB         COMF        1714 Sep 16  2004 .bashrc
-rw-rw-rw-   1 COMF.TB         COMF        3480 Aug 29  2007 .exrc
-rwxrwxrwx   1 COMF.TB         COMF         141 Jan 06  2008 .profile
-rw-------   1 COMF.TB         COMF         569 Jan 03  2007 .profile_fh
-rw-------   1 COMF.TB         COMF        1100 May 08 16:40 .sh_history
drwx------   1 COMF.TB         COMF        4096 Nov 02  2004 .ssh
-rw-------   1 COMF.TB         COMF        3116 Jan 08  2008 .viminfo
-rw-rw-rw-   1 COMF.TB         SUPER         15 Oct 20  2004 .vimrc
-rwxrwxrwx   1 COMF.TB         COMF       15000 Oct 24  2007 a.out
-rw-rw-rw-   1 SUPER.SUPER     SUPER    2722667 Aug 29  2007 abc
drwxrwxrwx   1 SUPER.SUPER     SUPER       4096 Oct 13  2004 bashtest
-rw-rw-rw-   1 COMF.TB         COMF         699 Oct 24  2007 block.c
-rwxr-xr-x   1 COMF.TB         COMF       27064 Jun 25 13:08 file0,0,1,1,1
-rwxrwxrwx   1 COMF.TB         COMF         244 Oct 24  2007 fixmore
drwxrwxrwx   1 COMF.TB         COMF        4096 Apr 25  2006 gnumisc
drwxrwxrwx   1 COMF.TB         COMF        4096 Jan 08  2008 hertz
-r-xr-xr-x   1 SUPER.SUPER     SUPER     389152 Mar 03  2005 ls
-rwxrwxrwx   1 COMF.TB         COMF         128 Mar 28 06:35 rc0071
```

```
-rwxrwxrwx   1 COMF.TB          COMF         126 Mar 28 06:36 rc0078
-rwxrwxrwx   1 COMF.TB          COMF         113 Mar 28 06:32 rc_bad
-rw-rw-rw-   1 COMF.TB          COMF         101 Nov 13  2007 resize.test
-rwxrwxrwx   1 COMF.TB          COMF          86 Mar 28 06:30 returncode_failure
-rwxrwxrwx   1 COMF.TB          COMF          80 Mar 28 06:31 returncode_success
drwxr-xr-x   1 COMF.TB          COMF        4096 May 03  2006 sshtest-client
drwxrwxrwx   1 COMF.TB          COMF        4096 Feb 18  2005 sshtest-daemon
-r--r--r--   1 COMF.TB          COMF        1000 Oct 08  2007 t1000
-r--r--r--   1 COMF.TB          COMF      100000 Oct 08  2007 t100000
-rw-rw-rw-   1 COMF.TB          COMF     1000000 Oct 13  2007 t1000000
drwxrwxrwx   1 COMF.TB          COMF        4096 Oct 13  2007 testdata
-r--------   1 COMF.TB          COMF         171 Mar 28 06:13 testfile1
-rw-rw-rw-   1 COMF.TB          COMF          13 Jan 18 12:33 testtail
-rw-rw-rw-   1 COMF.TB          COMF      100000 Oct 08  2007 tscroll
drwxrwxrwx   1 COMF.TB          COMF        4096 Dec 21  2005 tuxedo
-rw-rw-rw-   1 COMF.TB          COMF      533775 Feb 25  2005 zlib.tar.Z
226 Transfer Complete.
2674 bytes received in  0.45 seconds ( 5.80 Kbytes/s)
ftp> bye
221 Goodbye.
$TB TBSSH79 3>
```

The following log messages will show up in the SSH2 log file indicating that the session was indeed forwarded over the SSH session:

```
$TBS79|08Jul08 08:07:29.37|50|\NPNS01.$Z0DC: forwarding FTP connection from
127.0.0.1:1139 to 127.0.0.1:21
$TBS79|08Jul08 08:07:38.85|50|\NPNS01.$Z0DC: forwarding direct-tcpip connection from
127.0.0.1:1140 (accepted on 127.0.0.1:4518) to remote
$TBS79|08Jul08 08:07:44.32|50|\NPNS01.$Z0DC: closed forwarded FTP connection from
127.0.0.1:1139 to 127.0.0.1:21
```

SFTP Client Command Reference

The SFTP[OSS] Client is used to start interactive or batch file transfers from and to a remote system which are initiated from the NonStop system.

Command-Line Reference

The SFTP client allows you to specify some parameters on the command line. Starting the client without any parameters provides a syntax summary:

```
> sftposs
SFTPOSS client version T9999H06_22Jan2014_comForte_SFTPOSS_0097
missing parameter error (1,1):
usage: SFTPOSS [-vCZ] [-b batchfile] [-o ssh2_option]
            [-H error_prefix] [-J info_prefix] [-K query_prefix]
            [-B buffer_size] [-R num_requests] [-S ssh2 process]
            [user@]host[:file [file]]>
```

Note: The syntax for specifying local file names (files to be read or written on the NonStop system) supports both "Unix style" and "Guardian style". Please see the section "file name syntax" for details.

Runtime options

The following runtime options are supported:

-b <batchfile>

Starts the SFTP client in batch mode. The commands contained in the file are executed one by one until completion or a failure in execution. The client then terminates.

-B <buffer size>

Specify the size of the buffer that sftp uses when transferring files. Larger buffers require fewer round trips at the cost of higher memory consumption. The default is 29696 bytes (29kB). The maximum buffer size is 57344 bytes (56kB).

The transfer buffer size can also be set by specifying a PARAM/environment variable SFTPBUFFERSIZE.

-C

Requests compression of the transfer data. The compression algorithm is the same used by gzip. Compression is desirable on slow connections, but will only slow down the transfer on fast networks.

-o <ssh2 option>

Allows to pass an option for the ssh session to the SSH2 process

The following options are supported:

- BINDADDRESS=address

  The local address used for outgoing connections. Useful if the SSH2 process is configured with “any address” for parameter INTERFACEOUT or multiple IP addresses are configured in INTERFACEOUT, the TCP/IP process is configured with more than one subnet and a specific local address needs to be used (e.g. due to firewall configuration restrictions).

- IDENTITY=keyname

  Use this option to select a specific KEY for authentication to the remote system. By default all KEYs that you have generated using the SSHCOM GENERATE KEY command will be presented to the remote host for publickey authentication. However, some servers will deny authentication after a maximum number of inacceptable keys are presented, which can create a problem if you have many keys. To overcome this problem, use the IDENTITY option to present only the key that has been advertised as authorized key to the target server.

- PORT=port

  The port to connect to on the remote host.

- COMPRESSION=TRUE|FALSE

  Specify weather data compression should be enabled on the SSH session. This option has the same effect as the –C command line option.

- CIPHERS=ciphers

  Specify a comma-separated list of ciphers for encrypting the session.

- MACS=macs

  Specify a comma-separated list of MAC algorithms.

- USER=user

  Specify the user to log in as on the remote machine. This option has the same effect as specifying the user runtime parameter.

- AllowedAuthentications=*methods*

  Specify the authentication methods that are allowed for user authentication. The value is a comma separated list of method names (without any spaces). See SSH2 parameter CLIENTALLOWEDAUTHENTICATIONS for the possibility to restrict the sftp clients' authentication methods.

A typical usage of this option is to connect to an SSH2 daemon is running on a different port than the standard port 22:

```
> sftposs –oPort=2222 -S '$tba01' burgt@10.0.0.201
Connecting to 10.0.0.201...
sftp>
```

-R <num requests>

Specify how many requests may be outstanding at any one time. Increasing this may slightly improve file transfer speed but will increase memory usage. The default is 16 outstanding requests.

The number of outstanding requests can also be set by specifying a PARAM/environment variable SFTPNUMREQUESTS.

-S <SSH2 process name>

This option is used to set the SSH2 process to communicate with. Please refer to the section "Configuring the SSH2 Process to Use" earlier in this chapter.

-Z

The banner normally printed by the ssh client is suppressed (line " SFTPOSS client version T9999H06_23Dec2010_comForte_SFTPOSS_0089" in the above example). The suppression of the client banner can also be achieved by specifying a PARAM/environment variable SUPPRESSCLIENTBANNER with possible values 0 for false and 1 for true (the -Z option takes precedence over the PARAM/environment variable).

Runtime options relevant only when automating SFTP client

-H string

Set specific string used as prefix for error messages displayed by the SFTP client during the connection phase. Double quotes can be used to define strings containing a space or special characters. The prefix for errors can also be specified via PARAM/environment variable SSHERRORPREFIX (the -H option takes precedence over the PARAM/environment variable). There is no specific error prefix defined as default.

-J string

Set specific string used as prefix for informational or warning messages displayed by the SFTP client during the connection phase. Double quotes can be used to define strings containing a space or special characters. The prefix for infos/warnings can also be specified via PARAM/environment variable SSHINFOPREFIX (the -J option takes precedence over the PARAM/environment variable). There is no specific info/warning prefix defined as default.

-K string

Set specific string used as prefix for prompt/query messages displayed by the SFTP client during the connection phase. Double quotes can be used to define strings containing a space or special characters. The prefix for infos/warnings can also be specified via PARAM/ environment variable SSHQUERYPREFIX (the -K option takes precedence over the PARAM/environment variable). There is no specific query prefix defined as default.

Runtime Parameters

The following runtime parameters are supported:

User

The user name used to log on to the remote system.

Host

The IP address or DNS name of the host system to connect to. This parameter is mandatory.

File [file]

The remote file to download to the local system, optionally followed by the local filename of the downloaded file.

Examples for usage of runtime parameters

The following set of commands:

```
> sftposs -S '$TBA01' –oPort=2222 burgt@10.0.0.201
SFTPOSS client version T9999H06_22Jan2014_comForte_SFTPOSS_0097
Connecting to 10.0.0.201 via SSH2 process $TBA01 ...
```

```
sftp> help
Available commands:
ap     local-path [remote-path]  Upload local file and append to remote file
append local-path [remote-path]  Upload local file and append to remote file
ascii [dos|unix|mac]             Change transfer mode to ascii and optionally
                                 change the remote newline convention
aslinemode [cut|wrap|none]       Cut, wrap or do nothing to long ascii lines
binary                           Change the transfer mode to binary
cd path                          Change remote directory to 'path'
chgrp grp path                   Change group of file 'path' to 'grp'
chmod mode path                  Change permissions of file 'path' to 'mode'
chown own path                   Change owner of file 'path' to 'own'
delete path                      Delete remote file
exit                             Quit sftp
fc [<num>|<string>]              Fix command number <num> or contains <string>
get remote-path [local-path]     Download remote file
help                             Display this help text
h       [<cnt>]                  Display historic commands (all or <cnt> cmnds)
history [<cnt>]                  Display historic commands (all or <cnt> cmnds)
lap     remote-path [local-path] Download remote file and append to local file
lappend remote-path [local-path] Download remote file and append to local file
lcd path                         Change local directory to 'path'
lls [ls-options [path]]          Display local directory listing
lmkdir path                      Create local directory
ln oldpath newpath               Symlink remote file
lpwd                             Print local working directory
ls [path]                        Display remote directory listing
lumask umask                     Set local umask to 'umask'
mkdir path                       Create remote directory
progress [on|off|min|?]          Toggle display of progress meter (on/off) or set to
                                 minimum (value min) or display current setting
put local-path [remote-path]     Upload local file
pwd                              Display remote working directory
quit                             Quit sftp
rename oldpath newpath           Rename remote file
rm path                          Delete remote file
rmdir path                       Remove remote directory
symlink oldpath newpath          Symlink remote file
touch path                       Touch file
version                          Show SFTP version
?                                Synonym for help
sftp>
```

- Picks the SSH2 process $TBA01 to communicate with.
- Connects to the remote system with the IP address 10.0.0.201 on port 2222, using the user name "burgt".
- Uses the "help" command to show the commands supported by the SFTP client.

The following command:

```
/home/tb: sftposs -S '$tba01' burgt@10.0.0.201:a1000 testget
Connecting to 10.0.0.201...
Fetching /home/burgt/a1000 to testget
/home/burgt/a1000
100%  990     0.0KB/s   00:01
/home/tb:
```

- Picks the SSH2 process $TBA01 to communicate with.
- Connects to the remote system with the IP address 10.0.0.201 on port 2222, using the user name "burgt".
- Downloads the file "a1000" and places it locally under the file "testget".

Client Mode Owner Policy LOGINNAME The commands APPEND/LAPPEND do not support structured files.

SFTP Commands

Once you are connected to a remote system, the SFTP client issues a prompt "sftp>" and from then on supports the standard set of commands implemented in the SFTP protocol. The "help" command gives a brief syntax summary:

```
> run sftp -S $zss1 -oPort=51022 comf.us@10.0.0.196
SFTP client version T9999H06_22Jan2014_comForte_SFTP_0097
Connecting to 10.0.0.196 via SSH2 process $zss1 ...
sftp> help
Available commands:
ap     local-path [remote-path]  Upload local file and append to remote file
append local-path [remote-path]  Upload local file and append to remote file
ascii [dos|unix|mac]             Change transfer mode to ascii and optionally
                                 change the remote newline convention
aslinemode [cut|wrap|none]       Cut, wrap or do nothing to long ascii lines
binary                           Change the transfer mode to binary
cd path                          Change remote directory to 'path'
chgrp grp path                   Change group of file 'path' to 'grp'
chmod mode path                  Change permissions of file 'path' to 'mode'
chown own path                   Change owner of file 'path' to 'own'
delete path                      Delete remote file
exit                             Quit sftp
fc [<num>|<string>]              Fix command number <num> or contains <string>
get remote-path [local-path]     Download remote file
help                             Display this help text
h       [<cnt>]                  Display historic commands (all or <cnt> cmnds)
history [<cnt>]                  Display historic commands (all or <cnt> cmnds)
lap     remote-path [local-path] Download remote file and append to local file
lappend remote-path [local-path] Download remote file and append to local file
lcd path                         Change local directory to 'path'
ln oldpath newpath               Symlink remote file
lpwd                             Print local working directory
ls [path]                        Display remote directory listing
mkdir path                       Create remote directory
progress [on|off|min|?]          Toggle display of progress meter (on/off) or set to
                                 minimum (value min) or display current setting
put local-path [remote-path]     Upload local file
pwd                              Display remote working directory
quit                             Quit sftp
rename oldpath newpath           Rename remote file
rm path                          Delete remote file
rmdir path                       Remove remote directory
symlink oldpath newpath          Symlink remote file
touch path                       Touch file
version                          Show SFTP version
?                                Synonym for help
sftp>
```

Rather than going through each command in sequence, we will introduce the most important commands in a sample SFTP session in the next section

Sample Session

The following sample session shows some commands and how to use them.

The sample session shows usage of the SFTP client under OSS, however apart from starting the SFTP client from TACL rather than from the OSS shell, there are no differences in usage when running under TACL.

Start the SFTP client and connect to remote system:

```
/home/tb: sftposs -S '$tba01' burgt@10.0.0.201
Connecting to 10.0.0.201...
sftp>
```

Show current working directory on remote system:

```
sftp> pwd
Remote working directory: /home/burgt
sftp>
```

List files on remote system (detailed output):

```
sftp> ls -l
drwxr-xr-x    0 513      100          1200 Feb 11 15:10 .
drwxr-xr-x    0 0        0             608 Dec 31 12:04 ..
drwxr-xr-x    0 513      100            80 Feb 27  2004 public_html
drwxr-x---    0 513      100            48 Feb 27  2004 pubs
drwxr-xr-x    0 513      100            48 Feb  9 20:45 put
-rw-r--r--    0 513      100       1011018 Feb  9 20:40 putfiles
sftp>
```

Change to directory "put", list the files there (note that the directory is empty):

```
sftp> cd put
sftp> ls -l
drwxr-xr-x    0 513      100            72 Feb 14 07:31 .
drwxr-xr-x    0 513      100          1200 Feb 11 15:10 ..
sftp>
```

Show local working directory:

```
sftp> lpwd
Local working directory: /home/tb
sftp>
```

Verify the remote working directory:

```
sftp> pwd
Remote working directory: /home/burgt/put
sftp>
```

Transfer local file "a10000" to remote system:

```
sftp> put a10000
Uploading a10000 to /home/burgt/put/a10000
a10000
100% 9900     0.0KB/s   00:00
sftp>
```

List files on remote system (note the new file a10000):

```
sftp> ls -l
drwxr-xr-x    0 513      100            72 Feb 14 07:31 .
drwxr-xr-x    0 513      100          1200 Feb 11 15:10 ..
-rw-r--r--    0 513      100          9900 Feb 14 07:31 a10000
sftp>
```

Leave the SFTP client:

```
sftp> bye
/home/tb:
```

Transfer Progress Meter

SFTP/SFTPOSS client displays a progress indicator during file transfers if enabled. The progress meter can be enabled via command "progress on" and disabled via command "progress off". Entering the command progress without option will switch between the states "progress enabled" and "progress disabled".

If progress is disabled, the only line displayed for a download is "Fetching <remote-file> to <local-file>" and for an upload the line "Uploading <local-file> to <remote-file>" is shown.

In addition to option values on and off there is a third option "min" supported, which reduces the progress output to the last line: "<count> bytes transferred in <time> seconds (<rate>MB/s)".

Command "progress ?" will display the current setting (on, off, or min).

Controlling Transfer Summary

Summary information about each file transfer gets generated, e.g.:

```
165527760 bytes transferred in 86 seconds ( 1.8MB/s)
```

By default the number of bytes transferred is set to the EOF value of a file. This ensures consistency between the size of a file displayed by the ls -l command and the summary information. But the size of the actual content of a Guardian edit or structured file can differ greatly from the EOF value.

If it is of interest to see the actual number of bytes transferred in the transfer summary, then a define =SFTP^BYTES^TRANSFERRED can be set to ACTUAL:

```
ADD DEFINE =SFTP^BYTES^TRANSFERRED, CLASS MAP, FILE ACTUAL
```

The default value for this define is EOF, meaning the "bytes transferred" line contains the EOF value of a file in case the transfer was successful.

The define must exist in the environment of the SFTP[OSS] client.

Specifying File Names on the NonStop System

When specifying directories, subvolumes, or files on the NonStop™ system, the SSH2/SFTP implementation supports flexible ways to deal with the various notations:

- Files and directories under the OSS file system are specified using the normal Unix file name notation such as "/home/tb" for a directory and "/home/tb/myfile" for a file.
- Files and directories under the Guardian file system can be specified in two ways:
  - Using the normal Guardian notation, such as "$data1.tbhome" for a subvolume or "$data1.tbhome.myfile" for a file. Subvolume changes can be specified using the normal syntax such as "cd $data1.tbhome" or "cd mysubvol". Note that a subvolume needs to be present in a "cd" command.
    See the note below regarding Guardian file name notation.
  - Using the "Unix-style" notation for Guardian files. For instance, to specify the fully qualified file name "$data1.testvol.myfile", you can use the notation "/G/data1/testvol/myfile".

Note: Unlike with HP NonStop FTP, there is no explicit command ("quote oss" or "quote guardian") to switch between the two notations. The Guardian file name notation is only allowed if parameter SFTPALLOWGUARDIANCD is set to true, and if a "cd /G" command has first been issued to switch to the Guardian notation. The default for SFTPALLOWGUARDIANCD is false; for details, please refer to the description in chapter "SSH2 Parameter Reference".

Extended Syntax for Creation of New Guardian Files

By adding a comma and a list of options to a filename, the attributes for this file can be controlled in:

- "get" commands executed on the NonStop system.
- "put" commands executed on the remote system.

The syntax for get and put command is as follows, with the restriction that file attributes can only be appended to files in the Guardian name space:

```
get remote-file [ local-file [ ,file-attributes ]]
put local-file [ remote-file [ ,file-attributes ]]
```

where file-attributes is a comma-separated list, which contains different file attributes depending on file type.

For EDIT and unstructured binary files the file-attributes list is:

```
[ [filecode],[primary],[secondary],[maxextents] ]
```

For structured files the file-attributes list is as follows:

```
The [[filetype],[filecode],[primary],[secondary],[maxextents],
[record-len],[pri-key-len],[key-offset],[index-blk-len ]]
```

The file attributes, which must be specified exactly in the order shown above, are:

- filecode – the file code (integer from 0 through 32767)
- primary – primary extent size in pages (integer from 1 through 65535)
- secondary – secondary extent size in pages (integer from 1 through 65535)
- maxextents – maximum number of extents (integer from 1 through 978)
- filetype – file type indicator, e for an entry-sequenced file, k for a key-sequenced file and r for a relative file
- record-len – length of the records in a structured file
- pri-key-len – primary key length in a structured file
- key-offset – key offset in a structured file
- index-blk-len – index block length in a structured file

Examples:

- "get txe txe,700": will create a code 700 file
- "get bigfile bigfile,0,500,500,950": will create a file with ext (500,500) and maxextents 950
- "get keyseq keyseq,k,0,2,2,500,255,100,0,2048": will create a keysequenced file with ext(2,2), maxextents 500, recordlen 255, keylen 100, keyoff 0, blocklen 2048
- "get relative relative,r": will create a relative file
- "get entryseq entryseq,e": will create an entry sequenced file
- "get ascii editfile,101": will create a guardian edit file
- "put txe txe,700": will create a code 700 file
- "put bigfile bigfile,0,500,500,950": will create a file with ext (500,500) and maxextents 950
- "put keyseq keyseq,k,0,2,2,500,255,100,0,2048": will create a keysequenced file with ext(2,2), maxextents 500, recordlen 255, keylen 100, keyoff 0, blocklen 2048
- "put relative relative,r:" will create a relative file
- "put entryseq entryseq,e": will create an entry sequenced file
- "put ascii editfile,101": will create a guardian edit file
- "put bigedit bigedit,101,200,300,978": will create an edit file with ext (200,300) and maxextents 978

Refer to the *TCP/IP Applications and Utilities User Guide*, chapter "Communicating with the FTP Server", section "Transferring Structured Files" for a detailed description of this extended syntax.

The extended syntax can also be used in SCP commands.

Transfer Modes for Structured Guardian Files

The previous section described how to specify Guardian file attributes. This section introduce transfer modes, i.e. different ways to transfer structured files.

Per default, each logical record of a structured file is read and an end-of-record delimiter is added: LF ("\n") before the record is transferred. This transfer mode (delimited record transfer mode) corresponds to the FTP ASCII transfer of structured files (STRUCT R). Additionally, the following two transfer modes are supported: transparent transfer of records, and unstructured transfer of structured files.

The transparent transfer mode allows transferring records containing LF ("\n") characters inside a record. These files cause problems when being transferred in delimited record transfer mode as this character is used as end-of-record delimiter. This problem does not occur in transparent transfer mode but this mode can effectively be used for transfers from one NonStop server to another only (other SFTP implementations are not aware of the transparent mode implementation).

The unstructured transfer mode uses the Guardian option 'unstructured access of structured files' when opening a Guardian structured file. If the unstructured mode is enabled, SFTP and SFTPSERV read the structured file physically rather than logically (record by record). This transfer mode corresponds to the FTP BINARY transfer of structured files (STRUCT F). Files can only be read in unstructured transfer mode, i.e. if NonStop SFTP command 'put' is used or a remote sftp client issues a 'get' command against SFTPSERV on NonStop.

The transfer mode is specified by adding one of the following three characters after the file name, separated by a comma (no space allowed):

- D for delimited record transfer mode.
- T for transparent record transfer mode.
- U for unstructured transfer mode.

Examples:

1. A file named relseq1 needs to be read record by record, each transferred with the delimiter LF appended:

   ```
   sftp> get relseq1,d
   This is identical to
   sftp> get relseq1
   ```

   as transfer mode D is the default transfer mode.

2. An entry-sequenced file is to be transferred from a NonStop server to a Unix host:

   ```
   sftp> put entryseq,u entryseq
   ```

   The transfer mode and file attributes can be used at the same time; the transfer mode is appended to the file name first, then file attributes:

   ```
   <file>,<transfer-mode>,<file-attributes>
   ```

3. A key-sequenced file is transferred between NonStop systems:

   ```
   sftp> put keyseq,t  keyseq,t,k,541,128,128,16,4072
   ```

Transferring ASCII files

Both SFTP and SFTPOSS support transfers in ASCII mode. If ASCII mode is enabled, files will be automatically converted according to the server's newline convention for ASCII files. If required, the server's newline convention can be configured. Furthermore, if the target file is located in a Guardian subvolume, an edit file will be created automatically, without having to specify the file code explicitly in the file name.

The following commands control this feature:

- ascii [dos|unix|mac]
  changes to ASCII transfer mode and optionally sets the server's newline convention, where the meaning of the newline convention specifier is as follows:
  - dos: lines are terminated by a CR LF sequence ("\r\n")
  - unix: lines are terminated by a LF ("\n")
  - mac: lines are terminated by a CR ("\r")

- binary
  changes to binary transfer mode.

The following sample illustrates how ASCII files can exchanged with an SSH daemon on a Windows server:

```
sftp> ascii dos
Newline convention is now dos
File transfermode is now ascii
sftp> put textfile textfile.txt
Uploading textfile to /test/textfile.txt
sftp> get textfile.txt editfile
Fetching /test/textfile.txt to editfile
sftp>
```

In the above sample "editfile" is created as Guardian edit file (code 101), with the file correctly converted from the DOS ASCII format used by Windows.

When writing Guardian edit files SFTP and SFTPSERV convert TAB characters to spaces like FTP/FTPSERV if decimal line numbering is enabled (i.e. if parameter SFTPEDITLINESTARTDECIMALINCR is greater than or equal to 0 and parameter SFTPEDITLINENUMBERDECIMALINCR is not equal to 1000).

Fix Command and Command History

Within SFTP or SFTPOSS it is possible to list, modify and re-execute commands previously issued within the same SFTP or SFTPOSS session.

Command History

Historic commands are displayed when the HISTORY command is entered, e.g.:

```
sftp> history
   1> ls -l k*
   2> get file678
   3> put report89
   4> cd $disk.subvol
   5> cd $data1.reports
   6> pwd
sftp>
```

A maximum of 50 commands are saved. If only a smaller number of commands in the history list is of interest, a numeric parameter can be used to specify the number of commands, e.g.:

```
sftp> history 4
   1> ls –l k*
   2> get file678
   3> put report89
   4> cd $disk.subvol
sftp>
```

A string can be specified after the history command that controls the selection of historic lines: Only those lines of the history list are displayed that contain the supplied string, for example:

```
sftp> history t8
3> put report89
sftp>
```

History Mode

There are two different modes that can be set to manage the history list. The mode must be set via PARAM/environment variable HISTORYMODE before starting the SFTP[OSS] client, i.e. in the process environment of the SFTP[OSS] client.

Possible values for HISTORYMODE are SFTP (the default value) and TACL. If HISTORYMODE is set to TACL the history list behaves like the one in TACL.

The following table explains the differences between HISTORYMODE SFTP and TACL:

FC/HISTORY differences depending on HISTORYMODE setting	**HISTORYMODE SFTP**	**HISTORYMODE TACL**
Commands added to the history list	All commands but help, history, fc and "!"	All commands but fc and "!"
Default count for history command display	20	10
Handling of duplicate commands	Only the last of duplicate commands stays in list	Duplicate commands are added
Command number change	Command numbers change whenever an old, duplicate command is moved to the top	Command number assigned to a command stays the same until the command drops out of the history list
Match of string supplied as parameter to FC and HISTORY command	A string matches anywhere in a command line	A string must match the beginning of a command

Fix Command

The FC command (fix command) allows retrieving one of the history commands either by number or by string matching.

If a number is specified, then the corresponding command is retrieved and can be modified using standard fix command modifications via R, D and I (see “Guardian Procedure Calls Reference Manual”, section FIXSTRING for details).

```
sftp> fc 2
   get file678
...     d//i5
   get fle5678
...      r4// 9
   get fl456789
...
Couldn't stat remote file: No such file or directory
File "/G/data1/reports/fl456789" not found.
sftp>
```

If the FC command is followed by a negative number, then the corresponding command relative to the end of the history list is selected (-1 equates to last command, -2 equates to next-to-last command, etc.):

```
sftp> history
   1> cd $data1.reports
   2> dir
   3> get file1
   4> get file2
   5> get file3
   6> get file4
sftp> fc -3
   get file2
...//
sftp>
```

If a string is specified, then the corresponding command is retrieved using string matching, i.e. the last command containing the given string is retrieved and can be modified and executed.

```
sftp> fc rep
   cd $data1.reports
...                1
   cd $data1.report1
...
sftp> pwd
Remote working directory: /G/data1/report1
sftp>
```

It is possible to force string matching for a given number by enclosing the number in single or double quotes:

```
sftp> history
   1> ls -l k*
   2> get file678
   3> put report89
   4> cd $disk.subvol
   5> cd $data1.reports
   6> get fil56789
   7> get fl456789
   8> cd $data1.report1
   9> pwd
sftp> fc "4"
   get fl456789
...//
sftp>
```

The FC command without parameter causes the last command being retrieved for fix command processing.

A modified command is not executed (i.e. ignored) if the character sequence on the fix command line is ‘//’ as shown above.

The command “!<n>” to execute a command in the history list is not implemented. The following error is returned:

```
(‘!’) not supported for security reasons.
```

Creation of Format 2 Guardian Files

Since version 0092 it is possible to create format 2 files. In pre-0092 releases data could be read from and written to existing format 2 files but format 2 files could not be newly created during an SFTP session. Format 2 files had to be created before an SFTP transfer could write data to them.

The indication of a format 2 file is a plus sign directly appended to the file code of the Guardian file attributes, similar to the file code shown by FILEINFO for format files.

Examples:

```
sftp> get remote local,101+,28,56,128
sftp> put local remote,0+
```

Controlling SSH and SFTP Clients on NonStop via an API

Customers who need to access SSH and SFTP clients programmatically can use additional API modules, which are separately licensed:

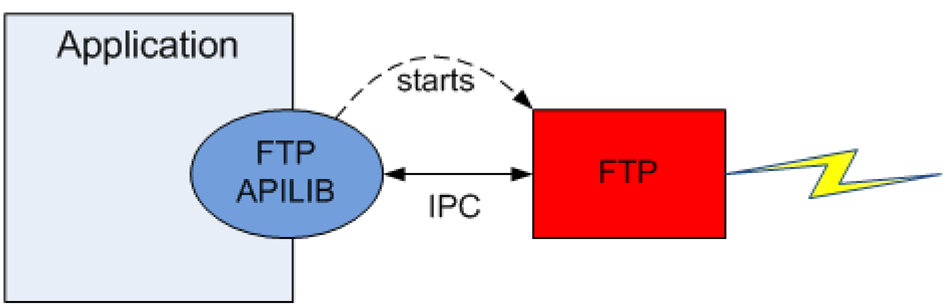
- The SFTPAPI module allows an FTPAPI application to establish an SFTP session instead of an FTP session. Minor changes in the FTPAPI application code converts the application to an SFTPAPI application. This is possible because the same header file ($SYSTEM.ZTCPIP.FTPEXTH) and library file ($SYSTEM.ZTCPIP.APILIB) is used as it is for FTPAPI.
- The SSHAPI/SSHLIB module provides a general way to access and control an SSH client on NonStop™ providing a means for automating tasks on a remote system or, when using loopback, on the local system.

The following sections give a short overview. For more detailed information see the *SFTP API Reference Manual* and the *SSHLIB Reference Manual*.

SFTPAPI

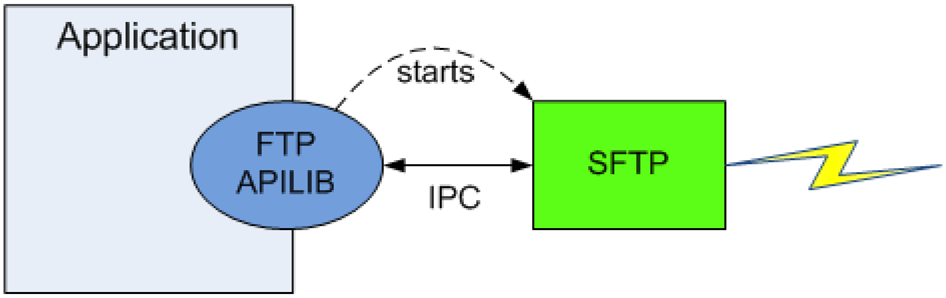
The SFTP API allows applications that previously used the FTP API to convert to SFTP in an easy manner. In many cases, the conversion can be accomplished with only a few program changes. In the ideal case, programs do not need to be changed or even re-compiled at all.

The following picture describes how applications transfer files with the FTP API:



When initiating an FTP session via the FTP APILIB, the library will start an FTP client process to handle the actual file transfers for the application. APILIB will then communicate via inter-process messages with the FTP client process, mapping the library calls to FTP commands to be processed by the FTP client.

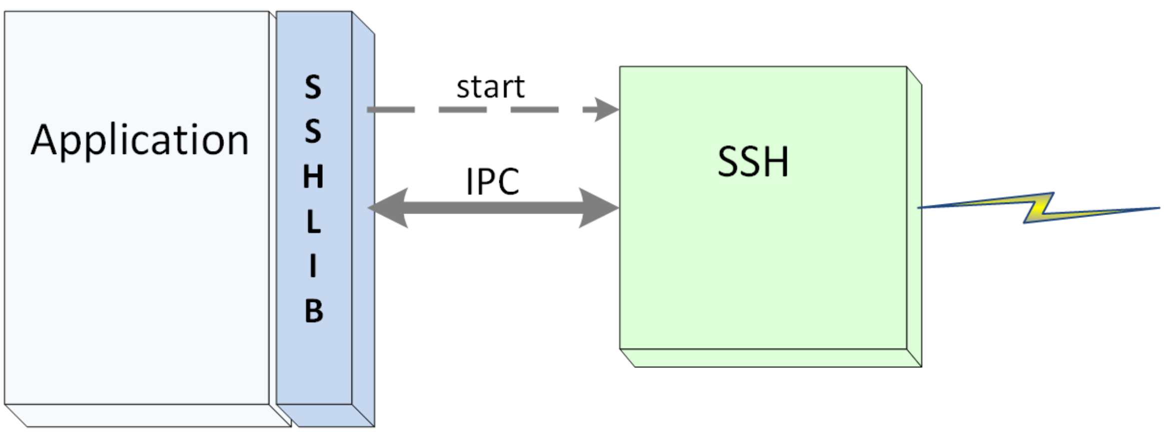
The SFTP API solution works exactly the same way, as the following picture illustrates:



For transferring files via SFTP rather than FTP, the application still uses the same APILIB, which is part of the HP NonStop TCP/IP applications and utilities. However, APILIB is directed to start an SFTP rather than an FTP client. The SFTP client will support the same inter-process communication messages like FTP, mapping the programmatic commands it to the appropriate SFTP operations.

SSHAPI with SSHLIB

SSHLIB describes the external interface offered by the SSH application program interface (API). SSHLIB is used for launching an SSH object and controlling it automatically by an application via the SSH API. SSHLIB can simplify the task of controlling status or resources on a remote host. It is also helpful to automate setup scripts for duplicating software package installations on different servers. There is no limitation for SSHLIB other than what the application developer can imagine regarding remote control tasks executed via an SSH session.



In the figure above it is depicted how the application communicates with SSH using SSHLIB. When initiating an SSH session via SSHLIB, the library will start an SSH process in SSH API server mode to handle the actual communication for the application. SSHLIB will then communicate via inter-process messages (IPC) with the SSH process, mapping the library calls to messages to be processed by SSH.

SSH will return required output and error information back to SSHLIB in the same fashion.

SSH Protocol Reference

The SSH Protocol

SSH is a protocol for encrypted network traffic and a set of associated programs which have its roots in the Unix domain. The first version of SSH (SSH version 1 or SSH1) became popular in 1995 and was replaced by an improved version (SSH version 2 or SSH) in 1997. In 2006, SSH version 2 became a proposed internet standard with the publication of a group of RFCs by the Internet Engineering Task Force (IETF).

For more information on the SSH protocol we recommend the following reading:

- "Secure Shell" in Wikipedia: http://en.wikipedia.org/wiki/Secure_shell
- A popular commercial SSH implementation for PC and Unix systems comes from a company called SSH. Their website is http://www.ssh.com.
- A guide to the generation of SSH key pairs can be found at http://apps.sourceforge.net/trac/sourceforge/wiki/SSH%20keys
- A comprehensive book on SSH is *SSH, The Secure Shell*, Daniel J. Barrett, published by O'Reilly

Implementation Overview

Supported Versions

The SSH2 software package only supports version 2 of the SSH implementation.

Cipher Suites

For a list of supported cipher suites and MACing algorithms, please see the parameters "CIPHERS" and "MACS" in chapter "Configuring and Running SSH2".

Implementation of the SSH protocol

SSH is a complex security protocol involving many sophisticated algorithms, therefore implementing SSH on any platform is not a trivial task. There are many intricacies in implementing SSH; just the fact that "it works" does not guarantee the quality of an implementation.

The following code has been used as part of the SSH2 software package:

- a commercial SSH implementation (bitvise sshlib, see http://www.bitvise.com/products.html) which is based on the popular crypto library crypto++ (see http://sourceforge.net/projects/cryptopp/).

- a small part of the OpenSSL project, see www.openssl.org.
- a small part of the OpenSSH project, see www.openssh.com.

comForte has combined this standard code with its own source code targeted specifically for the NonStop™ platform and has added additional functionality.

See the copyright statements in chapter "Appendix".

Authentication using User Names and Passwords

The SSH protocol allows for the authentication using user names and passwords. This mechanism is less secure than Public Key Authentication (discussed in the next section) and that is why most implementations allow to disable authentication using user names and passwords.

It is up to the SSH server to specify both the allowed and required means of authentication. comForte's SSH implementation currently supports the following means of authentication:

- When running as SSH client, the SSH2 package allows authentication using either a private key (configured using the KEY entity in the SSH2 user database, see next section) or a password (to be entered interactively or configured using the PASSWORD entity in the SSH2 user database)
- When running as SSH daemon, the SSH2 package currently supports both password (verified against the Guardian user password) and public key authentication (configured in the PUBLICKEY attribute of the USER entity of the SSH2 database)

Public Key Authentication

Introduction to Public Key Authentication, Terminology

Public Key Authentication makes use of asymmetric cryptography. Without going too much into details, we explain and define some terms here:

- A **key pair** consists of a public and a private key. While it is possible to derive the public key from the private key, the opposite is not possible.
- The **private key** is normally kept secret and can only be accessed by the entity using it for authentication. Among other things, a private key can be used for signing bits of information – without the private key nobody else can do this for a given key pair.
- The **public key** can be distributed freely as it contains only public information. Using the public key, documents signed using the private key can be checked for authenticity. When distributing public keys, it is important to make sure nobody has altered the public key during the distribution process.
- A **fingerprint** is a cryptographic "shorthand" for a public key. A public key basically is a set of bytes, however it is hard to compare a long stream of bytes. That is why fingerprints are used to verify public key. Two popular formats for fingerprints are MD5 (32 bytes of hex characters) and bubble-babble (16 words out of the "bubble-babble" word set).

The terms "key pair", "public key" and "private key" are all used to specify a key pair or a part of it.

Public Key Authentication and SSH

The SSH protocol uses public key cryptography for authentication both of the server (daemon) to the client as well as – optionally - for authenticating the client. This implies that if the client uses a key pair to log on to the server, both the client and the server will:

- have their own private key stored in the safe location
- send over the public key belonging to their private key to the peer system for authentication
- have the public key of the peer system configured in order to be able to verify its authenticity

Dealing with two key pairs for any two partners communicating can be a bit confusing, therefore we go over the two key pairs in a bit more detail in the next subsections. Please note that

- (A) when operating as SSH daemon, you are accessing your own private key and verifying the remote public key.
- (B) when operating as SSH client, you also are accessing your own private key and verifying the remote public key.
- the two key pairs mentioned under (A) and (B) are different resulting in a total of four key pairs being maintained when operating both as daemon and client. The following list shows all four key pairs and where they are configured in the comForte SSH implementation (the following subsections will go into a bit more detail, the names in brackets are repeated there for ease of reference):
    - (KEYPAIR1) A key pair used to authenticate the NonStop system to the partner system when the NonStop system acts as daemon (HOSTKEY parameter of SSH2 process)
    - (KEYPAIR2) A key pair used to log on the partner system to the NonStop system when the partner system is acting as client (PUBLICKEY property of USER entity in user database in daemon mode)
    - (KEYPAIR3) A key pair used to authenticate the partner system to the NonStop system when the partner system is acting as daemon (KNOWNHOST entity of user database in client mode)
    - (KEYPAIR4) A key pair used to log on a NonStop user on the partner system when the NonStop system acts as client (KEY entity of user database in client mode)

In the NonStop SSH2 implemention the local host key (KEYPAIR1 above) is of format DSA (1024 bit), the remote host keys (KEYPAIR3 above) can be DSA or RSA keys and the local or remote user keys (KEYPAIR4 and KEYPAIR2 above, respectively) can be DSA or RSA keys.

Assuring Host Authenticity

For every encryption protocol it is important for the client to check the servers authenticity. Not doing so enables the so-called man-in-the-middle attack which allows deciphering of the network traffic even though it is encrypted.

In the SSH protocol, authentication of the server is done by using public key authentication. The server generates a key pair; the private key of which he keeps to himself while sending the public key over to the client during connection setup. The client then verifies the public key and in order to be able to, the proper public key has to be configured at the client once.

Within the comForte implementation

- (KEYPAIR1) When acting as SSH daemon, the host key pair for the SSH2 daemon process is created during startup of the SSH2 process. It can be controlled with the "HOSTKEY" parameter described in chapter "Configuring And Running SSH2".
- (KEYPAIR3) When acting as SSH client, the public key of the remote host is configured by the KNOWNHOST entity of the user database.

Client logon

The client can also use a key pair to authenticate against the server; in this case the server will use that information instead of a password supplied by the client. The SSH protocol supports authentication of the client through various means:

- By providing a username and a password

- By providing a username and a public key
- By other means, such as Kerberos or X.509 certificates

When operating as a daemon, SSH2 currently supports the following authentication methods:

- password (RFC 4252)
  The password sent by the client is verified against the SYSTEM-USER's password contained in the NonStop system user base.
- Publickey (RFC 4252)
- keyboard-interactive (RFC 4256)
  The client is prompted for a password, which is verified against the SYSTEM-USER's password contained in the NonStop system user base.
- gssapi-with-mic, gssapi-keyex (RFC 4462)
  These methods are used for Kerberos authentication.

The same authentication methods are also supported when SSH2 is operating as a client. The following sections provide an overview of the publickey user authentication method.

Publickey client logon when operating as daemon

(KEYPAIR2) The public key of the client is configured in the user database with the PUBLICKEY FILE or PUBLICKEY FINGERPRINT property of a USER entity of the SSH user database. (please see chapter "The SSH User Database" for details).

To find out the fingerprint of an existing public key on a remote system, please refer to the documentation of the sftp implementation you use. The following example shows how to display the fingerprint with the ssh-keygen and the "-l" option utility in OpenSSH:

```
T:\>ssh-keygen -l
Enter file in which the key is (/home/comf.burgt/.ssh/id_rsa):
1024 5c:16:2f:95:fe:0e:1e:97:15:98:0f:ba:ae:32:c3:67 /home/comf.burgt/.ssh/id_rsa.pub

T:\>
```

The fingerprint to be configured on the NonStop system is highlighted in bold.

Publickey client logon when operating as client

The public key of the remote system is configured using the KNOWNHOST entity of the user database using the CLIENT mode of the SSHCOM command interpreter.

(KEYPAIR4) The private key used to log on the partner system is configured using the KEY entity of the user database using the CLIENT mode of the SSHCOM command interpreter. The public key to be configured on the remote system can be displayed using the INFO KEY command or exported into a file using the EXPORT KEY command.

STN Reference

Introduction

The STN component is a pseudo TTY server providing full-screen shell access to remote SSH clients.

Running STN as Pseudo TTY Server for SSH2

Note: For cases in which SSH2 was delivered with HP NonStop SSH as part of the RVU or as an independent product for G-Series prior to G06.32, an STN PTY server will be pre-installed as a generic process: SSH-ZPTY ($ZPTY).

Starting STN from TACL

STN can be started using standard TACL commands. It can also be configured as a generic process.

The example below shows how to start STN "from scratch", without a TACL routine:

```
1   logon super.super
2   volume $vol.subvol
3   clear all
4   param ...
5   run stn / name $PTY , pri 180 , nowait /
6   run stncom $ZPTY; ...
```

Following is a detailed explanation of each step:

1 - logon super.super

Like SSH2, the STN PTY server must be started under user SUPER.SUPER.

2 - volume $vol.subvol

Point to the subvolume where STN is installed.

3 - clear all

Clears all parameters for this tacl session.

4 – param …

Specify parameters. All parameters are optional. Except for TRACE^SIZE and TRACE^FILE, they may be specified in any order:

PARAM BACKUPCPU cpu

Specifies the backup CPU number. The default is NONE. See the STNCOM BACKUP/BACKUPCPU command for a description of available options.

PARAM GWN^TEMPLATE #AAAnnn

Controls session and window names. Refer to section "Session and Window Naming".

PARAM GWN^INITIAL RANDOM

Controls session and window names. Refer to section "Session and Window Naming".

PARAM GWN^FILE filename

Controls session and window names. Refer to section "Session and Window Naming".

PARAM GWN^BLOCKSIZE number

Controls session and window names. Refer to section "Session and Window Naming".

PARAM LICENSE filename

Specifies the location of the STN LICENSE file. The default is filename "LICENSE" in the subvol containing the STN object file.

Note that a license for NonStop SSH is no longer required, starting with SPR T0801^AAQ. STN does not require a license to run pty sessions with SSH. A license is required for optional features that are not available in NonStop SSH.

PARAM NOTACL 1

The value ("1" in the example) is not used; the presence of this PARAM disables the automatic default service TACL. If this parameter is NOT used, STN will automatically perform the command:

```
ADD SERVICE TACL,PROG $SYSTEM.SYSTEM.TACL
```

PARAM OPEN^TABLE^SIZE number

Specifies the maximum number of opens from application processes to STN windows. The default is 3000 and the maximum is 32000. See STNCOM command MAX^OPENERS.

PARAM POOL^SIZE number

Specifies the size in words of the extended segment memory pool used for control tables and I/O buffers. The default is 4194304 (4meg). A decimal number can be used to specify the parameter. Users may also append the letter K (kilowords) to the number, which multiplies by 1,024, or they can add the letter M (megawords), which multiplies by 1,048,576. POOL^SIZE may need to be increased for larger configurations; contact Support for details.

PARAM SECURITY letter

Defines the level of security access required for sensitive STNCOM commands. Sensitive commands are defined as commands that alter the STN environment. Non-sensitive commands are those that only report status information without changing anything in the STN environment. The default is O. Allowed values are from the set "NAGCOU" and are based on the standard Guardian file security interpretation.

PARAM TRACE^FILE trace-file

Starts a trace file immediately. The size is determined by PARAM TRACE^SIZE. This file is created if it does not already exist. The trace file must refer to a local disk file. PARAM TRACE^FILE should follow PARAM TRACE^SIZE. Tracing is normally started using STNCOM commands, so this parameter is rarely used.

PARAM TRACE^SIZE number

Specifies the byte size of the trace file when PARAM TRACE^FILE is used. A decimal number can be used to specify the parameter. Users may also append the letter K (kilowords) to the number, which multiplies by 1,024, or they can add the letter M (megawords), which multiplies by 1,048,576. The default is 100K. PARAM TRACE^SIZE should precede PARAM TRACE^FILE. Tracing is normally started using STNCOM commands, so this parameter is rarely used.

5 – run stn …

STN does not use the OUT parameter, example:

```
run stn / name $stn , out $zhome /  <--- not allowed
```

- If OUT is not defaulted to the home terminal, the following EMS event zstn-ems-evt-misc (9) is now generated:

  ```
  "$STN OUT parameter is not used, OUT <out> ignored."
  ```

  and STN startup continues normally.

STN uses the IN parameter to specify an edit-101 file. This file contains PARAM commands (other commands are ignored). Refer to the manual under GFTCOM^OBJECT and GFTCOM^IN for further details. The IN parameter may be used with or without PARAM GFTCOM^OBJECT.

- When IN is not specified, it defaults to the home terminal and STN startup continues normally without any "IN" processing.
- If the IN parameter specifies $ZHOME, the following EMS event zstn-ems-evt-misc (9) is now generated:

  ```
  "IN parameter must specify a edit-101 file or be omitted.  IN $ZHOME is ignored."
  ```

  and STN startup continues normally.
- If the IN parameter specifies something other than a disc file or $ZHOME, the following EMS event zstn-ems-evt-misc (9) is now generated:

  ```
  "IN file=<in> is not a disc file, startup terminated."
  ```

  and STN terminates abnormally.
- If the IN parameter specifies a disc file that is not an edit-101 file, the following EMS event zstn-ems-evt-misc (9) is now generated:

  ```
  "IN file=<in> is not a edit-101 file, startup terminated."
  ```

  and STN terminates abnormally.

STN does not use any parameters on the RUN command, including the backup cpu number in the manner used by other products. The STN backup cpu must be specified by either PARAM BACKUPCPU or the STNCOM command BACKUPCPU.

6 – run stncom …

Use stncom to enter additional configuration parameters and check settings.

Running STN as Persistent Process

STN can be started as kernel persistent process from SCF. The IN field of the RUN STN command is used to convey PARAM and STNCOM configuration information, as shown in the following example:

```
ADD PROCESS         STN1
              , NAME          $STN1
              , PROGRAM       $SYSTEM.STN.STN
              , INFILE        $SYSTEM.STN.STN1KIN
              , STARTMODE     SYSTEM     -or- APPLICATION
              , USERID        SUPER.SUPER
```

The INFILE (STN1KIN in this example) contains STNCOM commands to configure WINDOWs and SERVICEs, and also may contain PARAM commands as described above, but should always include the following:

```
PARAM GFTCOM^OBJECT $SYSTEM.STN.STNCOM
PARAM GFTCOM^IN $SYSTEM.STN.STN1KIN
PARAM GFTCOM^OUT $ZHOME
BANNER $SYSTEM.STN.BANNER1
```

STNCOM

STNCOM is the system operator interface to STN. STNCOM provides for configuration, status, and maintenance requests. You can store your STNCOM commands in an EDIT format disk file or enter them conversationally. You can direct your output to a terminal, printer, disk file, or spooler. Standard OBEY and FC commands are provided. A built-in HELP command is used; you can easily change the HELP dictionary or extend it to conform to local requirements by modifying the supplied STNCHELP EDIT file. When STNCOM is run, an implied OPEN $STN command is issued prior to prompting for input. STNCOM commands can be continued over multiple lines. When an ampersand ("&") appears as the last character on a line, the command is continued with the first column of the next line. There is no limit on the number of lines over which a command may be continued, but commands are limited to 10240 characters. Prior to STN version B24 the limit was 1024 characters. If STNCOM is prompting at a terminal for input, the prompt for continuation lines will be the current prompt prefixed by ampersand ampersand space: "&& ". Continuations are allowed from terminals, IN files and OBEY files.

Starting with version B08, responses to incorrect STNCOM commands will be preceded and followed by lines containing "*** Error ***".

To start STNCOM, use the standard TACL RUN command, as shown in the following examples:

```
1> RUN stncom $stn

2> stncom $stn1 ; info stn ; e

3> stncom / IN stnin4 , OUT $s /

4> stncom $stn1 ; TRACE $system.stn.trace3,1M ; e
```

The following illustrates a sample session:

```
STNCOM T0801H01_23JAN2012_ABA

OPEN $STN

%  info service

info service

SERVICE TACL

         TYPE        DYNAMIC

         PROG        $SYSTEM.SYSTEM.TACL

%  version

version
```

```
G007I \T.$STN 1,835
G000I STN B15 15NOV2011
G001I Copyright 1984-2011 Gemini Communications Inc.  All rights reserved.
%  exit
Exit
```

Starting with SPR T0801^ABE, the following banner and version info is displayed:

```
STNCOM T0801H01_24JAN2013_ABE
OPEN $STN
------------------------------------------------------------------------------
-  \T $STN   STN B21 04JAN2013   T0801H01_24JAN2013_ABE  14:55        -
------------------------------------------------------------------------------
% version
version

Version      STN B21 04JAN2013

Vproc         T0801H01_24JAN2013_ABE

Link gmt      04JAN2013_230358

Program object file     $QAHPSSH.T0801ABE.STN   type 800

Node           \T

Process       $STN 0,1164

Started at    2013-01-07 14:28

Time running            0d 0h 32m

Backup process          1,1175

Last backup takeover  no takeovers yet

% exit

Exit
```

Comments

It is possible to add comments in IN files, OBEY files and at the interactive prompt. Any text following an exclamation mark is treated as comment text. A comment line is continued on the next line if the last character is an ampersand.

Note: A single exclamation mark alone entered at the STNCOM terminal prompt means "repeat last command unchanged" while a single exclamation mark in an IN or OBEY file is treated as comment line.

STNCOM Commands

Note: STN is also delivered as component of comForte's SecurTN product, a fully functional, secure Telnet server. STN supports several commands and features related to the Telnet server functionality. For clarity, these commands and features are not part of this manual.

STNCOM supports the following abbreviated keywords in commands:

Command	Abbreviation
SERVICE	SER
SESSION	SESS
WINDOW	WIN

ABEND

Immediately stops the STN process, creating a ZZSA dump file. If STN is running with a backup, the backup will take over.

Use this command only on direction from support staff.

ABORT SERVICE

Same as STOP SERVICE.

ABORT SESSION

Same as STOP SESSION.

ABORT WINDOW

Same as STOP WINDOW.

ADD IPRANGE

Defines an IPRANGE for use with ADD SERVICE. Each IPRANGE defines 1 to 6 IP addresses or ranges of IP addresses.

```
ADD IPRANGE <iprange-name> <range> [,<range>]
```

<iprange-name>
1 to 8 characters, first alpha, remainder alpha or numeric, case insensitive. This name is used in the IPRANGE parameter of ADD SERVICE commands.

<range> has three allowable formats:

- a.b.c.d

  This form specifies a single IP address.

  Example:

  192.17.38.241

- a.*.*.*
  a.b.*.*
  a.b.c.*

  This form specifies the first 1, 2, or 3 bytes of an IP address which must match, with the remaining 3, 2, or 1 byte(s), respectively, allowed to have any value.

  192.*.*.*

  matches only 192.0.0.0 through 192.255.255.255

  192.7.*.*

  matches only 192.7.0.0 through 192.7.255.255

  161.114.87.*

  matches only 161.114.87.0 through 161.114.87.255

- a.b.c.d-e.f.g.h

  This form defines two specific IP addresses; the first must be numerically less than or equal to the second.

  192.1.2.3-192.1.2.6

  192.1.0.0-192.21.255.255

ADD IPRANGE command may be done before or after ADD SERVICE commands referring to the IPRANGE

ADD SCRIPT

```
ADD SCRIPT <script-name> function,p1,p2 function,p1,p2 …
```

A script is a series of setmode commands which is automatically performed at the beginning of a session and also after an application call to setmode 28. A script can be referenced by ADD SERVICE and ADD WINDOW commands. ADD SCRIPT and ADD SERVICE/WINDOW may be performed in any order, although the script must be defined before a session attempts to use it.

Example script to turn off echo and turn off automatic LF on CR:

```
ADD SCRIPT NOECHO 20,0 7,0
ADD SERVICE S123,SCRIPT NOECHO …
```

ADD SERVICE

The ADD SERVICE command defines a new service for STATIC and DYNAMIC window sessions. The service will be available to sessions on any LISTENER, as well as on SSH pseudo TTYs, if the CI-COMMAND *MENU* is set for the user as follows:

```
ADD SERVICE     service-name
    ,TYPE       DYNAMIC | STATIC
    ,PROG       program-file-name
    ,CPU        (cpunum | cpunum-cpunum | ANY)
    ,PRI        priority
    ,TERM_TYPE  TN6530 | ANSI | ANY
    ,MODE       BLOCK | CONV
    ,MENU       HIDDEN | VISIBLE
    ,LIB        lib-file-name
    ,SWAP       $volume-name
    ,USER       (groupnum,usernum) | groupname.username
```

```
,PARAM      "param-text"
,IPRANGE    iprange-name
,HOME       home-terminal-name
,LIMIT      max-sessions
,RESILIENT  YES | NO
,DEBUGOPT   OFF | <number>
,LOGAUDIT   YES | NO
,LOGON      REQ | NONE
,SCRIPT     script-name
,WIN_PAT   "pattern"
```

The service-name and the TYPE field are required; all others are optional.

TYPE	**DYNAMIC**	**STATIC**
CPU	optional	not allowed
DEBUGOPT	optional	not allowed
HOME	optional	not allowed
LIB	optional	not allowed
LIMIT	optional	not allowed
LOGON	optional	not allowed
PARAM	optional	not allowed
PRI	optional	not allowed
PROG	required	not allowed
RESILIENT	optional	not allowed
SWAP	optional	not allowed
USER	optional	not allowed
IPRANGE	optional	optional
LOGAUDIT	optional	optional
MENU	optional	optional
MODE	optional	optional
SCRIPT	optional	optional
TERM_TYPE	optional	optional

service-name

Service names are 1 to 8 characters long, beginning with a letter followed by letters and numbers. No special characters are allowed. Service names are always interpreted as upper case. The service name must not duplicate any existing services, including the default TACL service if present. The newly added service will be in a STARTed state and available for immediate use.

TYPE DYNAMIC

With TYPE DYNAMIC, the PROG field is required, while the CPU, PRI, LIB, SWAP, USER, PARAM, HOME, LIMIT, RESILIENT, DEBUGOPT and LOGON fields are optional.

When a session requests a dynamic service, a new window, with a unique name, is automatically created. A new application process is also automatically created. When the session terminates, the window is automatically deleted.

Dynamic services have various advantages and disadvantages:

- No WINDOW pre-configuration required.
- No application pre-configuration required.
- Workstations can have identical configurations.
- Unique window names are difficult to track and manage.
- Application process creation slows window startup.
- Can be awkward for Pathway and other applications that allocate CPU and other resources using their own algorithms.

Processes created by STN for SERVICE TYPE DYNAMIC that do not have a userid from LOGON REQ or from SSH authentication are started with CAID 0,0 (sometimes known as NULL.NULL) rather than 255,255 (SUPER.SUPER) as was done before version B20.

TYPE STATIC

The PROG, CPU, PRI, LIB, SWAP, PARAM, USER, HOME, LIMIT, RESILIENT, DEBUGOPT and LOGON fields are not allowed with TYPE STATIC.

When a session requests a static service, a search is made for a previously defined WINDOW that satisfies the following requirements:

- SERVICE field matches this service.
- TYPE is STATIC.
- Has an application running and waiting for a new session (CONTROL 11).
- Is not already in session.

If no such window is found, an error message is displayed and the service menu is repeated.

PROG program-file-name

Required when TYPE DYNAMIC is used; not allowed otherwise. PROG specifies the object file for the dynamic service to be started.

CPU (cpunum | cpunum-cpunum | ANY)

Default is (0,15) or as specified by DYN_CPU. Only allowed with the TYPE DYNAMIC parameter. Specifies the CPU number, or range of CPU numbers, in which STN will start the dynamic service application. If a range is specified, STN will "round-robin" each new session to spread the workload over the specified CPUs. ANY can be specified for any available cpu.

PRI priority

Only allowed with the TYPE DYNAMIC parameter. Specifies the process priority used to start the dynamic application. If omitted, the priority specified by the DYNAMIC_PRI command is used. Priority can be a number from 0 to 199.

TERM_TYPE TN6530 | 6530 | ANSI | ANY

TERM_TYPE controls the inclusion of services on STN02 Service menus. The default is ANY. TN6530 and 6530 are equivalent.

Workstation terminal emulators are divided into two groups. Those that support HP 6530 telnet extensions and which are configured for the HP 6530 protocol are considered type TN6530; all others are considered type ANSI. For TN6530 emulators, the STN02 will include only those services with TERM_TYPE TN6530 or ANY. For ANSI (all other) emulators, the STN02 will include only those services with TERM_TYPE ANSI or ANY.

TERM_TYPE only affects the display formatted for the STN02 Service menu. It does not restrict access to services or otherwise affect application or terminal activity. For example, an ANSI emulator could request a service configured for TERM_TYPE TN6530 even though the service name was not displayed on the STN02 service menu.

MODE CONV | BLOCK

Default is CONV. At the beginning of a session, the terminal (client) and the WINDOW are placed into the selected mode.

MENU HIDDEN | VISIBLE

Default is VISIBLE. Service menus are built using the names of services with MENU VISIBLE. MENU HIDDEN suppresses the service name on the menu, but the service name can still be entered by the remote user.

See the command "BANNER", which can disable menus and other messages.

LIB lib-file-name

Default is no LIB file. For dynamic sessions, this parameter specifies the library object file name for PROG program object files that require a library.

SWAP $volume-name

Default is no SWAP volume specified. Specifies the swap volume for dynamic sessions.

USER (groupnum,usernum) | groupname.username

USER is only allowed for TYPE DYNAMIC services. If USER is specified, it must match the userid authenticated for the session, or the session is terminated with an STN71 message. If the SSH userid has SYSTEM-USER not set to *NONE*, then that is the userid for the session; otherwise the userid and password are prompted from the terminal with STN15/STN16 messages. Whatever the source for the session userid, it must match the SERVICE USER parameter. USER is appropriate for applications which do not perform their own logon, or which need to be restricted. For example, RESILIENT services are often restricted to SUPER.SUPER.

As of STN version B17 (H06.25/J06.14), USER can be specified independent of LOGON REQ. Prior to that, when USER was specified, LOGON REQ was automatically set. When USER is present and LOGON is REQ, then the session must be authenticated for the specified userid, either by SSH or by response to the STN15 userid prompt. When USER is present and LOGON is NONE, the dynamic application will be started under the specified userid without authentication. Only use LOGON NONE when the application performs its own logon authentication.

PARAM “param-text”

Default is no parameter string. Allows the specification of a parameter string corresponding to the TACL command:

```
RUN program-file-name / NAME $pname ,... / param-text
```

Param-text is enclosed in double-quotes ("text"), it may be up to 100 characters long, and it may contain the following special characters:

- Two consecutive double-quotes ("") represent a single double-quote (").
- @W or @w is replaced by the window name e.g. $STN.#ZWN0001.
- @B or @b is replaced with the backup CPU number, which is the "buddy" of the CPU finally used for the dynamic application. The buddy of an even-numbered CPU is the next higher odd-numbered CPU, and the buddy of an odd-numbered CPU is the next lower even-numbered CPU.
- @I or @i is replaced with IP address of the client workstation.
- @S or @s is replaced by the security string returned by SSH, or "PLAIN" if the session is not secure.
- @@ is replaced with a single at (@).

IPRANGE iprange-name

iprange-name refers to a name of an IPRANGE (see ADD IPRANGE).

Default: None.

If no IPRANGE parameter is specified, then the service does not perform any checking on the IP address of the remote workstation attempting to connect to the service.

If IPRANGE is defined for the service, then the IP address of the remote workstation must match one of the IP addresses or IP address ranges in the specified IPRANGE. If the address matches, then the session is allowed to proceed. If the address does not match, or the IPRANGE is not defined, then the session is terminated ten seconds after displaying the following message on the remote workstation:

STN51 Workstation IP address not in range for requested service

Note that ADD SERVICE can be done before ADD IPRANGE; however any attempt to connect to the service will be rejected until the ADD IPRANGE command is completed. Similarly, DELETE IPRANGE will result in rejection of any connection attempts to services specified in the deleted IPRANGE until another ADD IPRANGE command is used to redefine the IPRANGE. If an ADD SERVICE command refers to an undefined IPRANGE, the ADD SERVICE command is accepted, and the following warning message is presented:

SERVICE added - warning IPRANGE not presently defined

HOME home-terminal-name

HOME controls the home terminal name for processes started by STN for TYPE DYNAMIC services. The default home terminal is the name of the dynamic window being started ($STN.#ZWNxxxx). If HOME is used, it should refer to a valid terminal name or to a home terminal process like $ZHOME.

HOME is needed in cases where a program continues to run after the STN session terminates. The most common example is when using the following configuration:

```
ADD SERVICE pathdyn,TYPE     DYNAMIC
                   ,PROGRAM  $system.system.pathcom
                   ,HOME     $zhome
                   ,PARAM    "$pm;run p65"
                   ,MODE     BLOCK
```

Without the HOME parameter, while the Pathway application starts and runs normally, a problem arises if the session is terminated from the workstation client. This results in PATHCOM creating a ZZSA dump file, usually in subvol $SYSTEM.SYSTEM.

LOGON REQ | NONE

LOGON controls user authentication for TYPE DYNAMIC services. The default is NONE, requiring no authentication before starting the application specified by PROG. This is appropriate when the application performs its own authentication, for example, TACL.

LOGON REQ requires authentication before starting the application. If the SSH SYSTEM-USER for the session is a valid Guardian userid, then that Guardian userid is used for the session. If SSH SYSTEM-USER is *NONE*, then STN will prompt the workstation user to enter a valid Guardian userid and password.

LOGON REQ should be used when PROG is the OSS shell (OSH).

LIMIT max-sessions

LIMIT controls the number of simultaneous sessions for a TYPE DYNAMIC service. The default is zero (0), which disables LIMIT and allows any number of sessions. Values 1-9999 may be specified. STN rejects any attempts to use a TYPE DYNAMIC service when LIMIT sessions are already active.

DEBUGOPT OFF | <number>

DEBUGOPT controls the debug-option parameter of Guardian procedure call process_create_, used when starting the application for TYPE DYNAMIC services. The default is OFF, which omits the parameter. A value in the range 0-7 is used to set the low order three bits <13:15> of the debug-option parameter.

Setting DEBUGOPT 0 will avoid a problem with PATHCOM leaving ZZSA files when a session is terminated at the remote workstation. Refer to ADD SERVICE parameter HOME for more information.

RESILIENT YES | NO

RESILIENT is an option for TYPE DYNAMIC services that allows the application to remain active after the terminal session is disconnected. The STN implementation of RESILIENT is similar in general functionality to that of HP Telserv, but with some key differences.

RESILIENT NO, the default setting, defines a traditional dynamic service. Upon session disconnect, file system errors are returned to the application, and most applications, like TACL, will detect this and stop. If KILL_DYNAMIC is set, STN will stop the application on session disconnect.

When RESILIENT is set to YES, LOGON is automatically set to REQ.

A typical use for RESILIENT is to define several TACL windows which run at high priority. By logging on to these TACLs once and disconnecting, they are primed and ready for quick reconnects. This avoids the overhead of process creation and logging on, which can be critical when a system administrator needs immediate access.

When a session requests a RESILIENT service, STN first checks for any existing windows left over from previous sessions for the service. If any such window is found, the session is connected to that window. The application that was running on that window during the previous session will, in general, repeat its prompt, but otherwise the session resumes exactly where it left off. For example, a TACL will still be logged on and have its environment intact. Specific operation during such a reconnection is described below:

1. STN first notifies the workstation user that the session is being reconnected to a resilient window with the message:

   ```
   STN70 Reconnecting to resilient window #ZWNnnnn
   Last access: <time>
   ```

2. Then STN displays information about any application programs running on the window, example:

   ```
   STN70 application $Y1G7 $SYSTEM.SYS00.TACL
   STN70 application 1,175 $SYSTEM.SYS00.FUP
   ```

   The application line is repeated for each opener of the window, including process name, cpu/pin, or posix pid, and the object file name. This helps clarify exactly what is running in the resumed session.

3. Finally the session is then resumed with handling dependent on the application I/O that was active when the previous session was disconnected.

   - ITI (conversational), read or writeread pending

     The application I/O is completed with febreak 111. For TACL and most other applications this repeats the prompt.

     For OSS (posix) reads, fesigint 4523 is returned. For /bin/sh and most other applications, this repeats the prompt.

   - ITI (conversational), no read or writeread pending

     This happens when TACL is PAUSE-d, etc.

     Guardian break or OSS SIGINT is generated, again generally resulting in a new prompt.

   - Block Mode 6530

     Terminal is placed into block mode. Error 191 is returned to the application. This forces most block mode applications to refresh the display.

     EDIT XVS will allow for session recovery. TEDIT refreshes the screen. Most Pathway applications refresh the screen.

If there are no existing windows, STN will create a new window and start a new application process, like any TYPE DYNAMIC service. The following message is displayed to clarify that a new session was created as opposed to a reconnect to a previous session:

```
STN70 No existing window available for resilient service,
```

```
window #ZWNnnnn added
```

When a RESILIENT session disconnects, there are certain differences from non-resilient dynamic sessions:

- No error code (140, 60, etc) is returned to the application, and no BREAK or SIGHUP sent. Any active application I/O request is left outstanding indefinitely. The application never notices that the session has disconnected.
- KILL_DYNAMIC does not apply.
- The window is not automatically deleted.

STN's implementation of RESILIENT differs from Telserv in the following ways:

- SERVICE TYPE DYNAMIC
- No ADD WINDOW command. Windows are dynamically created as needed. STN does not restrict a RESILIENT service to a single window, simplifying configuration.
- 6530 Block mode applications (EDIT XVS, TEDIT, Pathway) are handled cleanly.
- OSH (Posix) applications are handled cleanly.
- Multiple Guardian applications (for example, a FUP or SCF prompt started from a TACL) are handled cleanly.

LOGAUDIT YES | NO

LOGAUDIT YES is intended for PROGRAM $SYSTEM.SYSTEM.TACL, and will generate an AUDIT event when the TACL process first logs on. No additional event is generated if the TACL logs off, changes users, or if a second TACL process is started on the same terminal. Note that STN has a default ADD SERVICE TACL which has the default setting of LOGAUDIT NO, so to use this feature with the SERVICE named TACL, it is necessary to first DELETE SERVICE TACL to remove the default, then ADD SERVICE TACL,LOGAUDIT YES, etc to define a new service.

LOGAUDIT NO is default.

SCRIPT script-name

Default is no SCRIPT. Script-name refers to a list of setmodes defined by the ADD SCRIPT command. These setmodes will be performed at session initiation and whenever setmode 28 is performed by the application. ADD SCRIPT and ADD SERVICE can be specified in any order. If the SCRIPT is not defined, no error message is generated, and no setmodes are performed. ADD SCRIPT will take effect on the next session created for the service.

WIN_PAT "pattern"

Pattern must begin with a "#" (pound/hash sign), and the remainder must be letters, numbers, period, and substitution parameters. Except for substitution parameters, all other characters are copied directly to the window name, with letters being upshifted. Substitution parameters begin with at sign "@" followed by a letter and an optional width in parentheses "()". Parameter letters are case independent. Parameters marked GM are available only from Win6530 clients.

- @A - The group portion (before the ".") of the Guardian user name.
- @B - The user portion (after the ".") of the Guardian user name.
- @D - Date (LCT) in 8 digit format yyyymmdd
- @H - Client ip from TCP/IP, in fixed decimal format, twelve digits long E.G. 192.168.1.23 -> 192168001023
- @I - Client ip from TCP/IP, dotted decimal with dashes E.G. 192.168.1.23 -> 192-168-1-23
- @J - GMT juliantimestamp (micro secs in decimal format)
- @K - Client ip from TCP/IP, converted to hex without dots E.G. 192.168.1.23 -> C0A00117
- @L - The SSH process name without dollar "$".
- @P - STN process name (without $ prefix)
- @S - STN Service name

- @T - Time (LCT) in 6 digit format hhmmss
- @U - The external user name (alphabetic and numeric characters only).
- @X - STN expand node name (without \ prefix)
- @Y - STN expand node number

Substitution parameters @1 through @6 reference values returned by WSINFO. WSINFO is supported by Win6530 and some other terminal emulators. STNCOM WSINFO must be set to QUERY, REQUIRED or MATCH. Any fields not returned by the workstation are set to the null string. Only alpha and numeric characters are used; any others are discarded. Alpha characters are upshifted. For example, if the terminal reports "10.1.2.3" for the IP address field, then "@2" would yield "10123".

- @1 - Workstation "host" name
- @2 - Workstation IP address (which may be different from the value returned by @I due to NAT, firewalls, etc)
- @3 - Workstation domain name
- @4 - Workstation netBios name
- @5 - Workstation user name
- @6 - Workstation client name

Any parameter above may be followed by a width specification which is a number in round parentheses "()". A positive or unsigned number refers to the leftmost characters of the string, and a negative number refers to the rightmost characters. For example, assume the Expand node name is \PROD3:

```
@x       PROD3
@x(3)    PRO
@x(-1)   3
```

WIN_PAT defaults to "#ZWNnnnn" as with previous STN releases.

Example: Generate a name based on the last three bytes of the client IP address in hex:

```
WIN_PAT "#QPPW.QI@K[-6]"
```

an IP address of 10.18.127.163 would generate:

```
#QPPW.QI127FA3
```

If a window name is changed as a result of WIN_PAT, the following message will appear at the terminal

```
STN92 Window name changed from #ZWNnnnn to <new-name>
```

If the window name could not be changed because there was a problem in WIN_PAT, or because the new name duplicated existing window names, then the session is terminated after displaying the message

```
STN92 Window name change failed
```

Example configuration:

```
ADD SERVICE RESTACL,TYPE          DYNAMIC
                   ,RESILIENT     YES
                   ,PROGRAM       $SYSTEM.SYSTEM.TACL
                   ,MENU          HIDDEN
                   ,USER          SUPER.SUPER
                   ,PRI           199
                   ,LIMIT         3
```

Explanation of example settings:

MENU HIDDEN - this service is for use only by system administrators and only in case of emergency. General users won't see the service on the STN02 Services menu, avoiding confusion and minimizing undesired access attempts.

USER SUPER.SUPER - keeps unauthorized users away from this service, minimizes denial of service.

PRI 199 - high priority is sometimes essential for systems maintenance tasks, such as stopping a looping application.

LIMIT 3 - While only one window might be enough, allows extras "just in case".

LOGON REQ - (automatically set with RESILIENT YES) protects reconnection to previous sessions, and minimizes denial of service.

See INPUT_TIMEOUT for additional security that may be appropriate for resilient services.

ADD WINDOW

The ADD WINDOW command defines the file system access points that application programs are to use to exchange data with the remote terminal sessions. Prior to SPR T0801^ABE, ADD WINDOW was performed automatically for dynamic sessions when AUTO_ADD_WIN was enabled and an application open request was received for an undefined window. The AUTO_ADD_WIN configuration parameter is no longer supported. All openers of STN must refer to an existing window name.

```
ADD WINDOW        #window-name
    ,TYPE         DYNAMIC | STATIC | SU | DEDICATED
    ,TERM_TYPE    TN6530 | ANSI | ANY
    ,SERVICE      service-name
    ,IPADDR       dotted-ip-address
    ,SUBTYPE      nn | NONE
    ,SCRIPT       script-name
```

#window-name

This name uniquely identifies the window and, together with the $STN process name, is used by applications to exchange data with the remote terminal session. The name must be 2 to 8 characters long beginning with a pound sign (#) followed by a letter and optionally followed by letters or numbers. All letters are shifted to upper case.

When a window is automatically added for a dynamic session, a unique window name using the format #ZWNxxxx is generated, where xxxx is a unique number starting at 0000.

Starting with STN version B17, window names may now contain up to 16 characters following standard Guardian filename qualifier rules. Formerly, STN only allowed the first qualifer (the "middle" part of the file name $aaa.#MIDDLE); now STN also allows the second qualifier (the "third" part of the filename $aaa.#middle.THIRD). Case does not matter.

Examples:

#A

#B1

#def1234

#G.H

#J123456.k1234567

Note that only windows with one qualifer part (#A) may be specified in response to the Enter Choice> prompt. Windows with two qualifier parts (#B.C) cannot be specified in this way.

TYPE DYNAMIC

Normally used only internally by the dynamic window mechanism. SERVICE and TERM_TYPE are required, and IPADDR is not allowed. The window will be automatically deleted when the session terminates.

TYPE STATIC

SERVICE is required. IPADDR is not allowed.

Typically some number of static windows are defined for a given static service, creating a pool of windows to allocate to sessions requesting that service. Application programs must be pre-started before terminal sessions are allowed to access the service.

TYPE SU

SERVICE and IPADDR are not allowed. SU windows may only be accessed by specifying #window-name at the service menu, although they do not appear in the service menu in any form. SU windows allow a given terminal to connect to a specific window, which generally simplifies application configuration. A disadvantage is that each workstation must be configured to automatically select the unique #window-name, or the name must be manually entered. Having different configurations or procedures for each workstation presents logistical problems. See TYPE DEDICATED for an alternative.

TYPE DEDICATED

SERVICE is not allowed. IPADDR is required. DEDICATED windows are automatically connected when a session is started by a remote workstation with an IP address matching the IPADDR field. No service menu is displayed at all. This window cannot be connected by specifying #window-name at the service prompt.

DEDICATED windows allow the system manager to pre-configure all workstations in STN with their own window. Sessions from that workstation will always connect to the matching window, allowing precise control of application-window-workstation mapping. Unlike SU windows, the workstation configurations are identical, simplifying logistics.

TERM_TYPE TN6530 | ANSI | ANY

STN does not presently use the window TERM_TYPE setting.

SERVICE service-name

Not allowed with TYPE DEDICATED or SU; required with TYPE STATIC. Also required with TYPE DYNAMIC, but DYNAMIC windows are only internally created; they should not be entered via STNCOM.

For TYPE STATIC, this window is associated with the specified service name. This window can then be selected to satisfy session requests for the specified service.

IPADDR dotted-ip-address

Only allowed for TYPE DEDICATED. Specifies the IP address of the client workstation. Any session request from the specified IP address will be automatically connected to this window; no menu is displayed.

No two windows may have the same IP address. This means that remote nodes that want to run multiple sessions, especially terminal servers like AWAN 3883/4/5 or 3886 models, cannot effectively use TYPE DEDICATED.

SUBTYPE nn | NONE

Default is NONE. Otherwise a number in the range 0-63 may be used. See DEV_SUBTYPE command for details.

SCRIPT script-name

Default is no script. A script is a series of setmode commands which are automatically performed at the beginning of a session and also after an application call to setmode 28. A script can be referenced by ADD SERVICE and ADD WINDOW commands. ADD SCRIPT and ADD SERVICE/WINDOW may be performed in any order, although the script must be defined before a session attempts to use it.

AUDITCOLL OFF | <ems-collector>

AUDITCOLL names an EMS collector to receive EMS events for Audit-type events.

OFF is the default. No Audit-type EMS events are generated. Also used to stop generation of events.

Audit-type EMS events are written to the specified collector <ems-collector>.

AUDITCOLL specifies an EMS collector for "audit" EMS events (only). This is independent of $0 which always receives other EMS events.

$emscol is the name of an EMS collector which may specify $0 or an alternate collector.

AUDITCOLL OFF stops generating the new EMS events and closes the alternate collector (normal EMS events to $0 will continue in any case). See ZSTNDDL and ZSTNTMPL.

AUDITMSG <text>

Writes an audit event with the specified text.

AUTO_ADD_WIN DYNAMIC | STATIC | OFF

Starting with SPR T0801^ABE (STN version B21), the AUTO_ADD_WIN configuration parameter is no longer supported. All openers of STN must refer to an existing window name.

AUTODEL_WAIT <seconds>

Windows that are automatically added (TYPE DYNAMIC and AUTO_ADD_WIN) are automatically deleted when the TCP session is terminated or when all openers (applications) have closed the window. Some applications close the window and then quickly reopen it from a different process (this happens with Pathmon and Pathway TCP), this could prematurely delete the window. The AUTODEL_WAIT parameter allows a "grace" time that starts when the last opener closes the window. If another open occurs within the grace time, then the window and the session continue running. If the timer expires without any new opener, then the window is deleted.

The time given can be in the range from 0 to 20 seconds, the default is 3 seconds. A value of zero disables the feature, deleting the window immediately when the last opener closes.

Starting with SPR T0801^ABE, this command is not relevant with regard to AUTO_ADD_WIN since that parameter is no longer supported.

BACKUP[CPU] <cpu> | NONE | BUDDY | ANY | ?

BACKUPCPU controls the application backup process. BACKUP is a synonym for BACKUPCPU.

?
Displays the current setting, along with the current backup status.

NONE
Stops a backup process if one is already running. No new backup processes are created.

<cpu>
Specifies a number in the range 0 through 15 inclusive. The application will use the specified CPU for its backup process. If a backup process is already running, it is stopped. A new backup process is created in the specified CPU.

BUDDY
Toggles the low-order bit of the primary CPU number to determine the backup CPU number. This pairs CPUs for backup purposes in even-odd groups (0 to 1, 2 to 3, ...14 to 15). This avoids the problem of configuring a specific CPU number. If a backup process is already running, it is stopped. A new backup process is created in the specified CPU.

ANY
Uses any available CPU for the backup process. The first attempt is with the buddy CPU; if that fails, other CPUs are then used starting with CPU numbers closest to the primary until a backup is successfully started. This method assures that a backup will be created any time two CPUs are available. If a backup process is already running, it is stopped. A new backup process is created in the appropriate CPU.

BANNER Y | N

The BANNER command controls the display of menus on remote session initiation. The default is BANNER Y. When BANNER N is used to disable banners, no welcome messages or menus are displayed when a remote workstation connects to STN.

Note: BANNER N may interfere with 6530 emulators configured to automatically transmit the service name, or may interfere with emulator scripts.

BANNER_TIMEOUT <minutes>

BANNER_TIMEOUT allows for automatic termination of sessions waiting at the STN02 Service menu for an extended time. This releases resources used by idle connections.

BANNER_TIMEOUT 0, the default, disables the timeout. Sessions will not be terminated at the STN02 Services prompt.

The timeout can be specified in the range 3-14400 (3 minutes to 10 days). When the STN02 Service menu is unanswered for the specified length of time, the session is terminated.

If IDLE_WARNING is set to a non-zero value, then a warning message will be displayed once a minute when no input had been received, and fewer than IDLE_WARNING minutes remain until BANNER_TIMEOUT expires. The following message appears:

```
STN35 **WARNING** Terminal will be disconnected if it stays idle...
```

If input is received after this warning, the timer is reset and the session continues. If nothing is received when BANNER_TIMEOUT expires, then the following message appears:

```
STN36 Terminal was idle too long! Disconnecting...
```

This message will be displayed for approximately 10 seconds, then the session is disconnected.

The exact format of the STN35 and STN36 messages depends on the terminal type:

- 6530: Message is displayed at the cursor location and also on Line 25
- ANSI: Message is displayed at the cursor location

For services with LOGON REQ, the STN15 and STN16 messages prompt for a userid and password. If either of these prompts is not answered within 60 seconds, the session is terminated with an STN54 error message. This timeout always is in effect regardless of INPUT_TIMEOUT or BANNER_TIMEOUT.

See also:

- INPUT_TIMEOUT, IDLE_WARNING

BLAST <message>

BLAST <message> sends a broadcast to all active sessions.

<message> is limited to 54 characters of displayable ASCII (hex 20-7e). The text will be prefixed with BEL ESC o (hex 07 1b 6f) which will sound the audible beep and place the text on Line 25 for 6530 terminals.

This command should only be used for urgent messages since it can interrupt normal terminal activity.

BREAK_ON_DISCON Y|N

If this parameter is set to "Y", when a dynamic window session is disconnected, and there are no active I/O operations (e.g. WRITEREAD), a BREAK is simulated. No BREAK is sent if there is an active I/O. Default is "N".

BUFFER_SIZE

BUFFER_SIZE displays the size of internal STN buffers, which is useful in configuring STN memory via PARAM POOL^SIZE.

The BUFFER_SIZE command has no parameter.

C12_ALWAYS Y | N

C12_ALWAYS was introduced in STN version B22 (T0801^ABG) to modify control 12 (terminate session) application requests.

Y means control 12 requests always terminate the session regardless of the number of applications that currently have the terminal window open. Y is the default and is compatible with STN B21 and earlier releases.

N means control 12 requests are ignored unless there is only one remaining application open to the terminal window. Control 12 requests will only terminate the session when there is only one application open for the terminal window.

C12_ALWAYS should be set to N when one application starts another (which may in turn start yet another, etc), and control 12 requests from the secondary (etc) applications are to be ignored.

CHOICE_PROMPT Y | N

This command controls display of "Enter Choice> " prompt after the service name list. This is independent of BANNER Y|N.

Note: CHOICE_PROMPT N may interfere with 6530 emulators configured to automatically transmit the service name, or may interfere with emulator scripts.

CHOICE_TEXT "<text>"

Command CHOICE_TEXT can be used to redefine the Enter Choice> prompt which follows the STN02 Services menu.

<text> may contain any displayable ascii characters including space but excluding double quote ("), and may be from zero to 64 bytes long. <text> may contain "\N" or "\n" which will function as carriage return/line feed. Backslash followed by any other character will ignore the backslash and generate only the following character. The default is (notice the space at the end):

```
CHOICE_TEXT "\nEnter Choice> "
```

The setting is displayed by INFO PROCESS.

CONN_CLR_SSH Y | N

CONN_CLR_SSH controls clearing of the screen at connect time for SSH 6530 sessions. The clear occurs immediately before the STN00 message, which is after SSH BANNER and before STN WELCOME displays. Default is N, which is recommended with SSH BANNER Y, and is different from STN A91 and earlier. The current setting is displayed by INFO STN.

DELETE IPRANGE <iprange-name> | *

Deletes a specific IPRANGE or all IPRANGEs.

The IPRANGE is immediately deleted. If any SERVICEs refer to this IPRANGE, then those services will reject any new connection attempts until a subsequent ADD IPRANGE is done. In this case a warning is displayed in response to the DELETE IPRANGE command:

IPRANGE <name> deleted - Warning: 1 SERVICE(s) still reference this iprange

DELETE SCRIPT <script-name> | *

The specified script, or all scripts, will be removed from the configuration.

DELETE SERVICE <service-name> | *

The specified service, or all services, will be removed from the configuration.

DELETE WIN[DOW] <window-name> | *

DELETE WINDOW removes a previously added window from the configuration. Dynamic windows are automatically deleted upon session termination. Windows created by AUTO_ADD_WIN Y are automatically deleted when all applications using the window terminate or close the window (no longer relevant since SPR T0801^ABE where AUTO_ADD_WIN is not supported anymore).

WIN and WINDOW are equivalent.

<window-name> specifies a window to be deleted.

* means to delete all windows, including DYNAMIC and AUTO_ADD_WIN windows.

DEV_SUBTYPE   B05COMP | WINDOW | <nn>

Controls the values returned to an application that has called DEVICEINFO against a window.The following options are available:

```
B05COMP    (default) compatible with STN releases B05 and earlier.
            no session active       6,0
            6530 session active     6,4
            non 6530 session        6,0
WINDOW      response determined by ADD WINDOW configuration
            SUBTYPE nn              6,nn          (overrides TERM_TYPE)
            SUBTYPE NONE and no session active, response determined by TERM_TYPE:
               TERM_TYPE 6530       6,4
               TERM_TYPE other      6,0
            When SUBTYPE is NONE, and a session  is active, then B05COMP rules above are used.
<nn>        always responds with type 6 and subtype <nn>
```

DYNAMIC_PRI <nnn>

Specifies the default priority used for dynamic window applications when the SERVICE does not specify PRI.

Where <nnn> is the Guardian priority in the range 1-199; default is 149.

DYN_CPU (cpu,cpu)

Sets default CPU for subsequent ADD SERVICE TYPE DYNAMIC. Default is DYN_CPU (0,15).

DYN_WIN_MAX <nnn>

The existing DYN_WIN_MAX command is generally superseded by the features of GWN^TEMPLATE (introduced in T0801^ABE), but it is still allowed.

<nnn> is the maximum number of window names, including zero (0). <nnn> must be in the range 100 to 100000, default is 100000. DYN_WIN_MAX may be used to reduce the number of windows allowed by GWN^TEMPLATE. For example:

```
PARAM GWN^TEMPLATE #Z0000
STNCOM $STN ; DYN_WIN_MAX 250
```

cycles from #Z0000 to #Z0249, then back to #Z0000.

EXIT

EXIT stops STNCOM. This is the normal method of terminating an STNCOM session. STN is not affected. There are several forms of the EXIT command:

- EXIT
- E
- control Y
- eof on disc or process IN file

In an OBEY file, an eof command returns to the previous OBEY file or IN file, and does not terminate STNCOM.

FC

FC provides a typical FC facility; see Guardian TACL or EDIT documentation for a full description. Like the EDIT product's implementation, STNCOM allows FC to be combined with other commands on a line. When an FC command is combined in this manner, it takes effect after all other commands on the line are processed; then the FC applies to the entire line, including the FC itself. FC commands are not allowed in OBEY files, or when the IN file is not the same as the OUT file.

FESESSDOWN <error-code>

This command controls the file error code returned to application I/O requests while a session is down. Default is 140 (femodemerr) for compatibility with previous releases; values 10-9999 are allowed. Some applications expect error 66 (fedevdown) when a session is down.

FRAGSIZE <n>

Adjusts the minimum memory pool fragment size allowed when splitting a large buffer to satisfy a new request. Use only under direction of support staff. <n> can be in the range of 26 to 1000. If the larger buffer is within FRAGSIZE of the requested size, the buffer is not split. This can help reduce fragmentation of the buffer pool.

GWN [ALLOC]

STNCOM displays the GWN filename and details about the window name and option, and optionally a new block of names. This new command was introduced in T0801^ABE.

The following current information is always displayed:

```
GWN File name (or blank )
Blocksize
Next window name
Last window name allocated (same as next if no GWN File)
Maxmium window number
```

If ALLOC is specified, a new block of session names is allocated from GWN^FILE. Since allocation is normally done automatically, ALLOC is intended for development use only. Any window names reserved by a previous GWN^FILE allocation but not yet used are discarded. The next session will begin with the number just allocated.

HELP ALL | command

HELP provides online documentation to STNCOM users. The HELP file, named STNCHELP, is located in the same volume and subvolume as the STNCOM program object file. The file is in standard Guardian EDIT file format, with lines of text formatted according to certain rules. These rules are explained in comment lines within the STNCHELP file itself; list this file with EDIT or FUP for more documentation.

- HELP

  HELP without any parameters displays a summary of the HELP file.

- HELP ALL

  Displays all HELP information.

- HELP command

  Displays all HELP file information for the specified command.

IDLE_WARNING <n>

IDLE_WARNING controls the number of warning messages (one per minute) to be displayed before the session is terminated by INPUT_TIMEOUT or BANNER_TIMEOUT. <n> can be in the range from 0 to 14400. A value of zero (0) means no STN35 warnings will be displayed until the session is terminated with an STN36 message. The default is 2 (2 minutes).

INFO ALL

INFO ALL is a combination of INFO STN, INFO SCRIPT, INFO SERVICE, and INFO WIN. Only configured Windows are included, not Dynamic or PTY(SSH) windows. This command is useful when documenting STN configuration for support calls.

See also: SAVE_CFG.

INFO IPRANGE <iprange-name> | *

Displays configuration information for a specific IPRANGE or for all IPRANGEs.

INFO PROCESS

INFO PROCESS displays the setting of global parameters. The following example shows a typical result:

```
Config \BWNS02.$ZPTYK 075536 T0801H01_22JAN2014_ABK LG:18DEC2013_223018
 AUDITCOLL        OFF                  AUDITING         OPTIONAL
 AUTODEL_WAIT     3                    BANNER           Y
 BANNER_TIMEOUT   0                    BREAK_ON_DISCON  N
 C12_ALWAYS       Y                    CHOICE_PROMPT    Y
 CHOICE_ROW       0                    CONN_CLR_SSH     N
 CONN_CLR_TELNET  Y                    DEV_SUBTYPE      B05COMP
 DYNAMIC_PRI      149                  DYN_CPU          (0,15)
 DYN_WIN_MAX      100000               EMS_3270_CONN    OFF
 FESESSDOWN       140                  IDLE_WARNING     2
 INPUT_TIMEOUT    0                    KEEPALIVE        Y
 KILL_DYNAMIC     N                    LUNAME_ECHO      N
 MAX_OPENERS      32                   MAX_OUTQ         0
 NBOT             Y                    NBOT_TIMEOUT     8
 NEGOT_TIMEOUT    20                   NODE NAME        \BWNS02
 OPENER_WAIT      30                   OUTPUT_RESET     Y
 RECV_SIZE        1000                 REPLY_DELAY_MAX  2
 RFC860TM         0                    RSCMGR_DEPTH     3
 SEM2_SEGID_FIRST 900                  SEM2_SEGID_LAST  950
 SEM2_SEG_SIZE    130048               SEM2_SEND_MAX    4000
 SEM2_TIMEOUT     600                  SEND_LIMIT       30000
 SEND_Q_MAX       20                   SEND_TIMEOUT     120
 SPI              Y                    SSH_DEFAULT_SVC  *NONE*
 SSL_OBJECT       STNSSL               TERMID_RFID      N
 3270_IN_SIZE     2000                 3270_MORE_TO     5
 3270_SKIP_NEGOT  N                    3270_TM_BLOCK    10
 3270_TM_TO       2                    WELCOME_SEQ      BEFORE
 UAIPADDR         N                    WIN_AVAIL_C11    Y
 WIN_AVAIL_ALWAYS N                    WIN_SCRIPT_FIRST Y
 WSINFO           NONE
 CHOICE_TEXT      "\nEnter Choice> "
 STNCOM_PROMPT    ""
 TERMID_FILE      OFF
 WELCOME          OFF
  HP Build of STN, no license needed.  Allows umlimited PTY/SSH sessions
  SAFECOM INFO DISKFILE STN PRIV-LOGON    ON
  GWN disabled, using #ZWNnnnn for session/window names
  GWN^FILE
  GWN^BLOCKSIZE     25
  GWN prefix len    4
  GWN num digits    4
  GWN next window   #ZWN0001
  GWN last window   #ZWN0000
  SSL vproc         (none)
  SSH vproc         (none)
  Process Startup Params
PARAM BACKUPCPU      ANY
```

Note: Some commands displayed are not supported in HP T0801, for example CONN_CLR_TELNET and 3270_IN_SIZE. These commands are not documented in this manual and should not be used by HP T0801 users.

Comments

```
Config \BWNS02.$ZPTYE 075536 T0801H01_24JAN2013_ABE LG:04JAN2013_230358
```

Expand node name, STN process name, system serial number, STN vproc and LINKGMT.

```
SSH vproc         T9999H06_22Nov2010_comForte_SSH2_0089
```

This displays (none) until the first SSH session connects to STN, thereafter the VPROC of the SSH process.

```
Process Startup Params
```

If STN was started without params, displays

```
... no PARAMs ...
```

Otherwise, a list of PARAMs is shown, example:

```
PARAM BACKUPCPU            ANY
```

As of T0801^ABE, the GWN window and session parameters are displayed as well. See section "Session and Window Naming".

INFO SCRIPT <script-name> | *

Displays configuration information for the specified script or for all configured scripts.

INFO SER[VICE] <service-name> | *

Displays configuration information for the specified service or for all configured services. Only parameters which are different from the default are displayed.

Includes IPRANGE if configured for the service. Additionally, if the specified IPRANGE is not defined, a warning is displayed:

```
IPRANGE <name> - Warning: IPRANGE is not defined
```

```
%  info service
Info service
SERVICE TACL
                  TYPE        DYNAMIC
                  PROG        $SYSTEM.SYSTEM.TACL
```

INFO STN

Equivalent to INFO PROCESS.

INFO WIN[DOW] <window-name> | *

Displays configuration information for the specified window or for all configured windows. Only fields which are not set to default ADD WINDOW values are displayed.

If the window is connected to an SSH client, the command shows the following information:

```
% info win
info win
 #ZWN0001        TYPE        PTY
                             SCRIPT     PTY-SSH$
 #ZWN0002        TYPE        PTY
                             SCRIPT     PTY-SSH$

%  info win #zwn0001
info win #zwn0001
 #ZWN0001        TYPE        PTY
                             SCRIPT     PTY-SSH$
   pty command               pty-req
   vproc                     T9999H06_22Nov2010_comForte_SSH2_0089
   term_env_var              xterm
   term_rows                 24
   term_columns              80
   term_width                0
   term_height               0
   encoded terminal modes  03 00 00 00 7f 80 00 00 96 00 81 00 00 96 00 00
   client IP address         192.168.1.106
   client IP port            3839
   client channel            256
   external user name       SUPER.SUPER
   system user              SUPER.SUPER
```

```
auth method            keyboard-interactive
cipher                 aes256-cbc
mac                    hmac-sha1
compression            none
executed program       /bin/sh
kerberos principal     nam
local IP address       192.168.1.145
local IP port          22
TCP/IP process         $ZTCP5
```

The attributes have the following meaning:

- TYPE: The window type. PTY is displayed for windows allocated by an SSH2 process.
- pty-command: The command that the SSH2 process used to allocate the window.
- Vproc: The version of the SSH2 process that allocated the window.
- term_env_var, term_rows, term_columns, term_width, term_hight, encoded terminal modes: the client's terminal characteristic's passed in the SSH PTY allocation request
- Client IP address,- Client IP port :shows the remote IP address and remote port number of the SSH session.
- Client channel: Shows the SSH channel number of the terminal session.
- External user name: The user name that was used with SSH authentication.
- System user: The system user to which the external user name is mapped. *NONE* will be displayed if no system user is mapped.
- Auth method: The authentication method that was applied to authenticate the SSH user.
- Cipher: the encryption algorithm used on the SSH session.
- Mac: the message authentication algorithm used on the SSH session.
- Compression: Shows if data is compressed on the SSH session.
- Executed program: Shows any program started by an SSH2 process on that #window. The field is empty at the time of application startup, and is managed by STN (dynamic services) or externally (static windows).

INPUT_TIMEOUT <minutes>

INPUT_TIMEOUT allows for automatic termination of sessions that have been inactive for an extended time. This improves security and releases resources used by idle connections.

INPUT_TIMEOUT 0, the default, disables the timeout. Sessions will not be terminated due to inactivity.

INPUT_TIMEOUT <minutes> can specify a time in the range 3-14400 (3 minutes to 10 days). When the terminal is inactive for the specified length of time, the session is terminated.

The timer is always reset by terminal input (keyboard activity).

Note that for 6530 terminals which usually operate in line mode or in full screen (block) mode, simply typing a single character may not result in any transmission. To reset the timer, it may be necessary to use ENTER or a 6530 function key.

The timer can also be set by output activity from the application. If OUTPUT_RESET is set to Y, then application output will reset the timer the same as keyboard input. For example, an application that displays periodic output like an EMS console, would never timeout as long as it performed output at least once every INPUT_TIMEOUT minutes. If OUTPUT_RESET is set to N, then application output does not reset the timer, and keyboard input is required before INPUT_TIMEOUT expires.

If IDLE_WARNING is set to a non-zero value, then a warning message will be displayed once a minute when the terminal is idle, and fewer than IDLE_WARNING minutes remain until INPUT_TIMEOUT expires. The following message appears:

```
STN35 **WARNING** Terminal will be disconnected if it stays idle...
```

If terminal activity occurs after this warning, the timer is reset and the session continues. If the terminal is still idle when INPUT_TIMEOUT expires, then the following message appears:

```
STN36 Terminal was idle too long! Disconnecting...
```

This message will be displayed for approximately 10 seconds, then the session is disconnected.

The exact format of the STN35 and STN36 messages depends on the terminal type and mode:

6530 block mode	message is displayed on line 25
6530 conversational	message is displayed at the cursor location and also on Line 25
ANSI	message is displayed at the cursor location

See also: BANNER_TIMEOUT, OUTPUT_RESET, and IDLE_WARNING.

BANNER_TIMEOUT and INPUT_TIMEOUT can be used individually or in combination.

Note: For services with LOGON REQ, the STN15 and STN16 messages prompt for a userid and password. If either of these prompts is not answered within 60 seconds, the session is terminated with an STN54 error message. This timeout always is in effect regardless of INPUT_TIMEOUT or BANNER_TIMEOUT.

KILL_DYNAMIC Y|N

If set to "Y", when a dynamic window session is disconnected, the dynamically started process is stopped. Only a process directly started by STN would be stopped; descendant processes are not affected. Default is "N". In most cases the process will stop itself when it receives an I/O error on the STN window. Some applications do not stop immediately because they do not have an active read on the terminal. This command forces the immediate termination of the process.

LISTOPENS

Displays one line for each OPEN of the application by another process. Example output lines:

```
1.  G083I process.term [cpu,pin] fnum userid programfile home [backup]
2.  1 $TCP1.#W742 1,47 fn=6 id=20,33 $SYSTEM.SYSTEM.PATHTCP $TERM4 bak=2,52 fn=6
3.  2 \CENTDIV.01,050.#COMMAND.COMMAND fn=3 id=255,255 $SYSTEM.SYSTEM.STNCOM $OSP
```

These three example output lines represent the following:

1. Title line
2. Indicates that:

   OTX (open table index)=1. Each opener has an entry in the open table. The named process $TCP1 (cpu,pin=1,47) has opened the application with a terminal name of #W742 as file number 6.

   $TCP1's process access ID is group,user=20,33

   $TCP1's object program file name is $SYSTEM.SYSTEM.PATHTCP

   $TCP1's home terminal is $TERM4

   $TCP1's backup process (cpu,pin=2,52) has checkopened the application with file number 6
3. Indicates that:

OTX (open table index)=2. Each opener has an entry in the open table. The unnamed process running on node \CENTDIV with cpu,pin=1,50 has opened the application with terminal name #COMMAND.COMMAND as file number 3.

The #COMMAND.COMMAND terminal name indicates a STNCOM requester.

The program is running under group,user=255,255 (SUPER.SUPER) from object program file name $SYSTEM.SYSTEM.STNCOM with home terminal $OSP.

Note: the LISTOPENS command can generate a very long response.

MAX_OPENERS <n>

Defines the maximum number of application openers of a window. <n> may be in the range 1-512 and defaults to 32. Prior to STN version B22, the allowed range was 1-64. Any open attempts beyond the maximum will be rejected with feopenstop 61. This feature prevents an ill-behaved application from monopolizing STN resources. Larger values of MAX_OPENERS may require an increase in PARAM OPEN^TABLE^SIZE, especially when many windows are active.

MAX_OUTQ <n>

MAX_OUTQ defines the maximum number of messages queued for a window. Default 0 (zero) means no maximum. Allowable range is 0-50. If the limit is exceeded (by an unusual application), an EMS message is generated and the session is terminated. Use only on recommendation of HP support staff.

NBOT Y|N

STN supports Non-Blocking OSS Terminals (NBOT) which is used by the Posix system call select(). The NBOT command can be used to disable this feature. The default "Y" enables NBOT by setting bit<11> in the misc flags field in replies to Posix open messages. NBOT N clears bit<11> to indicate select() is not supported, to be compatible with STN releases prior to B08.

NBOT_TIMEOUT <seconds>

NBOT_TIMEOUT controls error recovery for NBOT. The default setting is 8 (seconds). When NBOT=Y, if STN cannot open or writeread a select ready message to Terminal Helper ($ZTTnn), after NBOT_TIMEOUT seconds STN will send a Posix SIGQUIT (control-\) to the application. Setting NBOT_TIMEOUT to 0 (zero) disables the feature, usually meaning the application will hang until Terminal Helper finally responds. The signal can occur promptly after NBOT_TIMEOUT expires, but can be delayed as much as 60 seconds.

NEGOT_TIMEOUT <seconds>

This is the time allowed for IAC negotiations to complete, defaulting to 20 seconds. If the timeout expires, usually due to the TN6530 client improperly configured with line mode disabled, an STN50 message is displayed for 10 seconds, then the session is terminated. <seconds> can be in the range from 1 to 120.

OBEY <edit-file-name>

OBEY processes STNCOM commands from an EDIT format file.

<edit-file-name> specifies the EDIT file in which the commands are listed. Commands can be nested up to six levels deep.

OPEN <STN-process-name>

OPEN opens the specified STN process for subsequent commands.

<STN-process-name> specifies the process to be opened. If another process is already open, that process is closed. If the OPEN fails, all STNCOM commands requiring an application are rejected until a successful OPEN is completed. The STN version and vproc are displayed after a successful OPEN, before the STNCOM prompt.

Examples:

```
OPEN $STN
```

```
OPEN $STN2
```

OPENER_WAIT <seconds>

OPENER_WAIT specifies a timeout at the beginning of the session while waiting for the application to first open the window. OPENER_WAIT allows values from 1-300 (1 second to 5 minutes) and defaults to 30 (seconds). Note that AUTODEL_WAIT formerly performed this function, but has been changed as described.

If no application opens the window, after OPENER_WAIT seconds, the screen will be erased (for 6530 terminals) and the following message appears:

```
STN38 No application program active on this terminal for nnn seconds. Session terminated.
```

This message will be displayed for several seconds, then the session will be terminated.

OPENER_WAIT now also applies to dynamic window sessions which before release A83 had a fixed wait time of 60 seconds. For this case, the existing error message STN41 is used.

OUT <filename> | STOP

STOP

Output to home teminal

<filename>

If a disc file that does not exist, it is created as file code 101 unstructured and is written as an edit-101 file.

If an existing unstructured disc file with code 101, it is erased and written as an edit-101 file.

If an existing disc file that is not unstructured or not code 101, or a non disc file, then the file is opened and sent lines of output.

OUTPUT_RESET Y | N

Determines if INPUT_TIMEOUT applies to sessions that have ongoing output, even if there is no keyboard input. When OUTPUT_RESET=Y, any application output to a terminal resets the timer just as if input was received from the terminal. This means that a terminal that regularly updates the display, such as an EMS or console log, may never time out. When OUTPUT_RESET=N, then INPUT_TIMEOUT applies even if output is being displayed, giving additional security. Default is Y. See also INPUT_TIMEOUT.

PAUSE

PAUSE suspends the STNCOM prompt. Use BREAK to return to the STNCOM prompt.

POOL

POOL verifies the integrity of STN's internal buffer pool and provides useful information for tuning PARAM POOL^SIZE.

```
POOL
```

- TOTAL SIZE—Shows word size of pool.
- IN USE—Shows words currently in use in the user buffer area.
- HIGH—Shows the highest value of IN USE since process startup or the most recent backup takeover.
- GETS—Shows total number of buffer allocation requests.
- PUTS—Shows total number of buffer releases.
- REJECTS—Shows the number of requests that failed due to pool exhaustion or fragmentation.
- TRIMS—Shows the number of trims (where a large buffer is allocated and the unneeded trailing portion is released while the front part is still used).
- BUFS IN USE—Shows number of buffers allocated, not yet released. HIGH specifies the highest value of BUFS IN USE.
- $RECEIVE msgs—Shows total user data and system messages on $RECEIVE.
- BYTES RCVD—Shows total bytes read on $RECEIVE.
- BYTES REPLIED—Shows total bytes replied to $RECEIVE.
- FRAGMENTS -- Shows number of fragments
- FRAGSIZE -- Shows size of fragment

PROMPT "<text>"

This command redefines the prompt sent to the terminal for new STNCOM input. It is also available in SSHCOM.

<text> may contain any displayable character except quote ("), and may be 1 to 64 characters long. Certain embedded commands (case independent) in <text> are replaced as follows:

- $P – the target process name
- $X – the target expand node name
- $T – target system LCT time in format HH:MM
- $D – target system LCT date in format yyyy/mm/dd
- $N – ascii carriage return line feed. This allows for multi-line prompts including blank lines.
- $B – ascii bel character which some terminal emulators will sound as a beep tone.

Example:

```
PROMPT "$X.$P $D $T STN> "
\DEV.$STN2 2010/08/06 23:59 STN>
PROMPT "$T $P> "
23:59 $STN2>
```

The default setting is PROMPT "% ".

The PROMPT command remains in effect until STNCOM terminates. The null string ("") can be specified to disable a previously entered prompt string.

If it is desired to retain the prompt across STNCOM sessions, command STNCOM_PROMPT should be used. See the description for STNCOM_PROMPT for more details.

PTY_REPLY_LEN <n>

Byte length of reply from STN to SSH.

<n> can be in the range from 1 to 16384. Default is 4096.

RECV_SIZE <nnn>

Specifies the byte length of socket receive buffers used to accept incoming session data. <nnn> is in the range 100-4095, default 1000. Larger values offer some improvement in performance, but only when large input messages are common. Smaller values conserve buffers in the memory pool which may be necessary with a large number of simultaneous sessions.

REPLY_DELAY_MAX <seconds>

This command sets the maximum delay time, in seconds, for an STN reply to an I/O error. An I/O error is defined as application I/O to the terminal (read, write, etc) which results in an STN reply with non-zero fecode (140, 110, etc.). This protects against poorly coded applications that hard-loop on I/O errors, consuming a cpu.

The reply to the first I/O error after a normal I/O is not delayed; the second consecutive error is delayed for 0.01 second. The delay time is multiplied by 4 for successive errors up to REPLY_DELAY_MAX seconds. The first time this limit is reached for a session, the following EMS event is generated:

```
zstn-evt-application-loop 1018
<stn-proc> <appl-proc> <progfile> is looping on window
<#window>
```

Example:

```
$ZPTY \T.$X1G4 $SYSTEM.SYSTEM.TACL is looping on window
#ZWN0001
```

REPLY_DELAY_MAX defaults to 2 seconds, and values from 1 to 60 are allowed. REPLY_DELAY_MAX 0 disables the feature, which means a looping application and STN can consume 100% of a cpu.

RESET SERVICE <service-name> | *

This command will reset the cumulative sessions counter to zero. Note that this is the only counter affected by RESET. Also note that RESET does not default to "*" like INFO and STATUS; to reset counters for all services, RESET SERVICE * is required, not just RESET SERVICE.

RSCMGR_DEPTH <n>

Specifies the number of simultaneous Resource Managers internal to STN. The range is 1 to 25, default 3. The Resource Manager handles dynamic sessions and logon processing, including the creation of the dynamic application. If all Resource Managers are busy, new dynamic session requests can be delayed. When the rate of new dynamic session requests is very high, performance can be improved by increasing RSCMGR_DEPTH. Use only under guidance from HP support staff.

SAVECFG <filename>

SAVECFG creates an edit-101 text file containing the current STN configuration. This is useful for configuration management and for generating complete documentation for support cases.

SAVECFG also includes commentary information about the STN process. SAVECFG deals only with STN, and does not include SSH configuration information.

If the file already exists, it is purged. A new file is created.

The file will contain commands suitable for direct input to STNCOM, including process parameters such as IDLE_TIMEOUT and WELCOME, as well as ADD commands for services, windows (types STATIC, SU, and DEDICATED only), scripts, and ip ranges. ADD commands will span multiple lines using "&" (ampersand) as a continuation character, so STNCOM T0801H01_24JAN2011_AAS, T0801G06_15DEC2010_AAT, or later, is required to accept the commands in the SAVECFG output file.

SECURITY [<letter>]

SECURITY displays and modifies the application's security setting. This setting is initially established by the PARAM SECURITY command, with a default of O.

If the parameter is omitted, the current setting is displayed. The value O is the default. The letter entered sets the associated level of security. Users can choose from “NACGUO” selections, which are based on standard Guardian file security interpretation. These letters assign access as follows:

- N—Any local or remote user
- A—Any local user
- G—A group member or owner
- C—A member of the owner's community (local or remote user with the same group ID as the owner)
- O—The owner only
- U—A member of the owner's user class (local or remote user with the same user ID as the owner)

The SECURITY letter controls access to sensitive commands by STNCOM users. Sensitive commands are defined as commands that alter the STN configuration or operation. Sensitive commands can only be performed by STNCOM users with a user ID matching the SECURITY setting. Non-sensitive commands, such as STATUS, INFO, and LISTOPENS, can be performed by any user ID.

SHUTDOWN

SHUTDOWN initiates an STN process termination, which takes about three seconds. All active sessions are terminated. There are no parameters.

You can also use the TACL STOP $STN-process-name command, but this can result in some warning messages.

SPI Y | N

This command can be used to disable SPI support. Default Y is compatible with pre-B18 releases. If set to N, opens to $STN.#ZSPI will be rejected with fenosuchdev (14), and if there is already a #ZSPI open, any future I/O requests will be rejected with fenusuchdev (14). This command is intended for Development use and should only be used under direction of support staff.

SSH_DEFAULT_SVC <service-name> | *NONE*

SSH_DEFAULT_SVC defines a default service to be used when the SSH userid is configured with CI_PROGRAM *MENU* (without anything following *MENU*). If SSH_DEFAULT_SVC is set to *NONE*, the default value, then the STN02 service menu is displayed and the user must type in the service name or #SU window name. If SSH_DEFAULT_SVC is set to any other value, then it is used as a service name and an STN73 message notifies the user of this fact.

START SERVICE <service-name> | *

Activates a service previously STOPPED or ABORTED. New session requests for the service will be accepted. START is automatically performed by ADD SERVICE, and is generally not used.

START WINDOW <#window-name> | *

Activates a window previously STOPPED. New session requests for the window will be accepted. START is automatically performed by ADD WINDOW, and is generally not used.

STATUS SERVICE [<service-name> | *]

Displays current status information for the specified service or for all services.

The output has the following format:

```
SERVICE <name>   <status>, Cumulative sessions=<a>
                , WINDOWs: Configured=<b>, In session=<c>, Available=<d>
```

<status>
STARTED or STOPPED.

<a>
Total number of sessions ever connected to this service.

<b>
Number of windows presently configured for this service.

<c>
Number of currently open sessions for this service.

<d>
For static services, the number of windows with application opens, ready for new sessions.

STATUS SESSION [<session-name> | *]

STATUS SESSION shows all active sessions, even those that have not yet been attached to a window. The output format for sessions created via SSH is as follows:

```
<window> <state> <terminal-info> <age>
```

<window>
The window name associated with this session. During session startup, this can refer to a dynamic window that has not yet been created. For static windows, this name will be changed to the static window name.

<terminal-info>
TT Terminal Type, for instance TN6530-8.
M Mode, for instance 6530-Line.

<age>
The age of the session in seconds

<state>
Tracks the progress of a new session.

- NEGOT

  Telnet IAC negotiations are in process with an SSH 6530 client.

- NEGOT_LM

For TN6530 sessions, line mode has been established, and the STN is waiting for TERMTYPE. This state usually lasts for less than a second.

- NEGOT_TT

  For TN6530 sessions, TERMTYPE has been established, and the STN is waiting for line mode. This state usually lasts for less than a second.

- MENU_NEEDED

  TERMTYPE has been established, and, for TN6530, line mode has been established. This state is usually immediately replaced by MENU.

- RESIL_RECON

  A resilient window has been reconnected to a new session. This state is usually immediately replaced by CONNECTED.

- MENU

  STN is waiting for a service name (or window name) from the remote SSH 6530 client, usually after displaying a menu of service names.

- ABORTED

  The session has been aborted, but is being left up for a short time to allow the user at the remote SSH 6530 client to notice and read error messages that describe the reason for session termination.

- RSCMGR_BUSY

  For dynamic sessions, all resource managers are presently busy with other new dynamic session requests.

- DYN_PROC_LAUNCH

  For dynamic sessions, the associated process is being launched.

- DYN_PROC_OPEN_TO

  For dynamic sessions, the associated process is being opened to write the startup message.

- DYN_PROC_SUMSG

  For dynamic sessions, the startup message is being written to the associated process.

- DYN_PROC_OPEN_FROM

  For dynamic sessions, the associated process has been launched and has received the startup message, but has not yet opened the STN window.

- CONNECTED

  The session is connected to a window. If a service is associated with the session, its name is displayed.

- PTY_INIT

  An SSH2 process has created the pseudo terminal (PTY) under its control. Any application processes on the terminal are started by SSH2.

STATUS WINDOW [<#window-name> | *]

Displays current status information for the specified windows or for all windows. The output format for sessions created via SSH is as follows:

```
<window> <status> <a> openers
```

```
<param-list>
```

<window>
Window name e.g. #ZWN0002.

<status>
STARTED (not in session), STOPPED, or IN SESSION.

<a>
Indicates that either "no" or "1 or more" applications have this window open.

<param-list>
Detailed information such as term_rows, term_columns, client IP address, etc.

STIX [RESET]

Displays cumulative statistics on the number of sessions. STIX displays the counters; STIX RESET displays then resets.

STNCOM_PROMPT "<text>"

This command redefines the prompt sent by STNCOM to the terminal for new command input.

<text> may contain any displayable character except quote ("), and may be 0 to 60 characters long. Zero means to use the default STNCOM prompt. Certain embedded commands (case independent) in <text> are replaced as follows:

- $P – the target process name
- $X – the target expand node name
- $T – the target system LCT time in format HH:MM
- $D – the target system LCT date in format yyyy/mm/dd
- $N – ascii carriage return line feed. This allows for multi-line prompts including blank lines.
- $B – ascii bel character which some terminal emulators will sound as a beep tone.

Example:

```
STNCOM_PROMPT "$X.$P $T STN> "
\DEV.$STN2 2010/08/06 23:59 STN>
STNCOM_PROMPT "$T $P stncom> "
23:59 $STN2 stncom>
```

The default setting is STNCOM_PROMPT ""

PROMPT and STNCOM_PROMPT are related commands. They both change the prompt used for STNCOM commands, and both allow parameter substitution such as $P for process name. But they take effect in different ways. PROMPT affects only the current STNCOM process execution, and is cancelled when STNCOM stops. Other STNCOM users are not affected. STNCOM_PROMPT setting is saved in the memory of the running STN process. It takes effect on all subsequent STNCOM openers of the STN process.

When STNCOM starts the default prompt string for conversational command input is percent space ("%"). STNCOM then opens the STN process specified in RUN STNCOM <process-name>. If the STN process has STNCOM_PROMPT configured, it will be used for the prompt. This will stay in effect until another OPEN command or until a PROMPT command.

PROMPT

- Redefines the prompt for the current STNCOM process execution.
- Takes effect immediately unless an STNCOM_PROMPT is in effect.
- Does not affect other STNCOM users.
- Must be entered every time STNCOM is used, which is inconvenient.

- Is overridden by STNCOM_PROMPT.

STNCOM_PROMPT

- Redefines the prompt for all future STNCOM openers to an STN process.
- Does not take effect until the next STNCOM open (see note below).
- Is stored in the configuration of the running STN process, which is convenient.
- Is maintained on a backup takeover of STN.
- Must be re-entered every time STN is started.
- Overrides PROMPT.
- Is included in SAVECFG output.

STNCOM_PROMPT is normally included in the OBEY or IN file used to configure STN at STN process startup, if a prompt other than the default is desired. However, if STNCOM_PROMPT is manually entered from a conversational STNCOM session, it does not take immediate effect. However, INFO STN will show the new setting. To force immediate use of the new setting, either stop and restart STNCOM, or use the OPEN command to reopen the same STN process. The new STNCOM_PROMPT setting will then be used by STNCOM.

STNCOM_PROMPT Example:

```
11> stncom $ZPTY
STNCOM T0801H01_22JAN2014_ABK - 2014-01-24 15:25:12.354
OPEN $zpty
-------------------------------------------------------------------------------
-  \BWNS02 $ZPTY   STN B25 18DEC2013   T0801H01_22JAN2014_ABK  15:25        -
-------------------------------------------------------------------------------
% info stn
.. .STNCOM_PROMPT    "" ...
% stncom_prompt "$P % "
stncom_prompt "$P % "
Accepted
% info stn
...
 CHOICE_TEXT      "\nEnter Choice> "
 STNCOM_PROMPT    "$P % "
 WELCOME          OFF
...
% time
time
19Dec11 09:15:46.35
% open $zpty
open $zpty
$ZPTY % time
time
19Dec11 09:16:09.71
$ZPTY %
```

STNLOG <text>

Provides a means to enter log messages to the STN EMS output. It is intended for the SSL process to send log output, generally for fatal errors, to a central location. The syntax is as follows:

<text> is any text up to 128 characters long. Generally not used from STNCOM.

STOP SERVICE <service-name> | *

The specified service, or all configured services, will be marked as stopped. The service name will not be displayed on menus, and will be rejected if entered in response to the service prompt. Use START SERVICE to resume the service. Existing sessions will not be affected. This command is not normally used.

STOP SESSION <session-name> | *

The specified session, or all active sessions, will be terminated.

STOP WINDOW <#window-name> | *

The specified window, or all configured windows, will be stopped. If a session is active on the window, it will be immediately terminated. Dynamic windows and automatically added windows will be deleted. The window will no longer be available for new sessions. Use START WINDOW to resume normal operation. This command is not normally used.

TIME

Displays the current date and time.

TRACE

This command controls writing of a trace to a disk file. The GTRED utility that is distributed in the SSH subvolume can be used to format the trace:

```
GTRED / in <trace-file> , OUT <list-file> /
```

GTRED formats EMS events recorded in the trace file using Guardian procedure EMSTEXT. EMSTEXT by default uses the system template file, which may not contain the latest STN templates which are provided in the STN release subvol file ZSTNTMPL. To use templates from an alternate location, use the same DEFINE as is used by EMSDIST before running GTRED:

```
delete define =_ems_templates

add define =_EMS_TEMPLATES,FILE

$SYSTEM.STNB20.ZSTNTMPL
```

The TRACE command has the following syntax:

```
TRACE { ? | OFF | RESET | [ON] filename [ ,size] }
```

?
Displays the current status and setting of the trace file and all parameters.

OFF
Stops the trace.

RESET
Resets the trace file pointers, effectively restarting the trace, but without the overhead of closing and reopening the trace file.

ON filename [,size]
Starts a trace on the specified unstructured disk file. The filename should be fully qualified; if it is not qualified, the default volume and subvolume in effect at the time the STN application was started are used, not the defaults from the STNCOM startup. If the file name does NOT begin with $ or \, the keyword ON is required. A file of the specified size will be created. If a trace is already open, it is first closed. The trace file can specify the same name as an already active trace file. In that case, the trace file is rewritten. The TRACE RESET command is more efficient for this purpose. Size determines the byte length of the trace file. The number can be followed by the letter K (kilobytes) which multiplies by 1,024, or the letter M (megabytes) which multiplies by 1,048,576. The default is 100K. The minimum is 12K and the maximum is 25M.

Starting with STN version B20, STN trace files are secured "OOOO" and CLEARONPURGE to better protect any sensitive data. Trace files, which are created by explicit STNCOM command or a PARAM at STN startup, contain all data to and from the remote terminals, including sensitive data like passwords. Even when SSL or SSH encryption is used to protect the data in motion, the data is unencrypted in trace files. Always follow best practices with trace files.

Starting with STN version B08, trace files will include INFO STN output at the beginning.

Warning: Tracing can noticeably affect response time and CPU usage.

UAIPADDR Y | N

STNCOM command UAIPADDR controls the inclusion of the workstation remote IP address on USER_AUTHENTICATE_ calls. This IP address is included in certain Safeguard records. UAIPADDR should only be used on Guardian releases H06.26 (or later) or J06.15 (or later). Using the parameter on earlier releases will cause an abend of the STN process and a ZZSA dump file created in the STN object file subvol. STN formerly used PROCESSOR_GETINFOLIST_ items 3 and 60 to retrieve the Guardian version number, but in certain cases the reported version number can be incorrect, leading to an STN abend.

UAIPADDR N (default) omits the IP address on USER_AUTHENTICATE_ calls. Safeguard records will not include the IP address. This can safely be used on all Guardian releases.

UAIPADDR Y includes the IP address on all USER_AUTHENTICATE_ calls without regard for the Guardian version. Safeguard records will include the IP address. If UAIPADDR Y is used on Guardian releases earlier than H06.26 or J06.15, STN will abend.

Notes:

- On Guardian Gxx releases (S-Series hardware), STN never includes the IP address on USER_AUTHENTICATE_ calls regardless of UAIPADDR setting.
- STN only calls USER_AUTHENTICATE_ for (a) SSH sessions configured with *MENU* and (b) Telnet sessions using a SERVICE with LOGON REQ.

VERSION

VERSION displays the process name and cpu, pin, revision number and revision date of STN. There are no parameters.

Starting with SPR T0801^ABE the following items are displayed: Version, Vproc, Link gmt (build timestamp), Program object file name and type, Node name, Process name and cpu,pin, process start time, Time running, Backup cpu,pin, Time of last backup takeover, and number of takeovers.

WELCOME <filename> | OFF | LIST

Displays the contents of an edit file to be displayed at session startup before the STN02 Services menu.

<filename>
Loads specified edit-101 file as welcome text. Text is limited to displayable ascii characters (hex 20-7e), 79 columns per line, and 50 lines. The text is saved in STN memory and the file is closed.

OFF
Turns off welcome

LIST
Displays current welcome text

INFO STN will show the status of WELCOME, but not the text.

WELCOME_SEQ BEFORE | AFTER | BOTH

WELCOME_SEQ controls the sequence of the WELCOME display relative to the Enter Choice> prompt. The default setting is BEFORE, which displays the WELCOME text before the Enter Choice> prompt. AFTER displays the WELCOME text after the response to the Enter Choice> prompt. BOTH displays the WELCOME in both places.

WIN_AVAIL_ALWAYS Y | N

Controls availability of dedicated windows to connect to a new session. Default N means availability is determined by WIN_AVAIL_C11. When set to Y, a DEDICATED window is always available for connection to a new remote session request, even if there is no active open from any application to that window.

WIN_AVAIL_C11 Y | N

Determines availability of a window when a static service is selected from the STN02 menu, or a session attempts to connect to a dedicated window. Set to Y, the window is available if one (or more) control 11 requests are outstanding. The default is Y. Set to N, the window is available if the window has one (or more) application openers. If the window is available the session is connected to it; if not, STN13 error message is displayed followed by a repeat of the STN02 service menu.

WSINFO NONE | QUERY | REQUIRED | MATCH

The command WSINFO requests workstation information using ESC-9e supported by the Win6530 and J6530 emulators by comForte. The information fields HOST NAME, IP ADDRESS, and USER NAME are retrieved and displayed in the STATUS SESSION command.

INFO STN displays the current WSINFO setting

The various values of WSINFO work as follows:

NONE
Nothing is sent to the workstation—this is default behavior.

QUERY
ESC-9e is sent to WS after the first response to the Service prompt (or at the equivalent time for TYPE DEDICATED windows). STN will wait five seconds for a response. The response is included in a new AUDIT event and is shown by STATUS SESS. The session always continues regardless of the response of even if no response is received.

REQUIRED
Like above, but a response is required. If none is received, the session is terminated with the following message displayed on the Workstation for 10 seconds:

```
STN57 This 6530 emulator does not support required WSINFO.
```

MATCH
Like above, but in addition, the IPADDRESS in the response must match the network IP address from accept_nw, or the session is terminated with the following message displayed on the Workstation for 10 seconds:

```
STN58 WSINFO address does not match network address.
```

WINSCRIPT_FIRST Y | N

Since release A74, all SSH windows are automatically configured with a SCRIPT PTY-SSH$. If this script was defined by ADD SCRIPT, then the specified setmodes were performed, otherwise no setmodes were done. However, this did not allow any script specified for a SERVICE to apply to SSH sessions.

WIN_SCRIPT_FIRST now allows SSH sessions to use the script defined for the selected service.

Y
The default for compatibility with B19 and earlier releases. SSH sessions either use script PTY-SSH$ if configured or if PTY-SSH$ is not configured, then no script. Any script defined with the service used for the session is ignored.

N
SSH sessions use the script, if any, defined for the service. If none is defined, then the script defined for the window, if any, is used; otherwise, no script. This allows SSH sessions to access STN services which specify their own scripts.

The current setting is shown by INFO STN.

Session and Window Naming

Session (and dynamic window) names always began at 0000 when STN was started. This resulted in the same session name being used for different STN processes or for restarts of an STN process. The session names should be unique.

Starting with SPR T0801^ABE, a new optional naming scheme was introduced for sessions and dynamic windows. The default still uses names like #ZWN0001.

A related new feature provides for the pooling of window names over multiple STN processes, and over restarts of STN processes.

PARAM GWN^TEMPLATE #AAAnnn

GWN^TEMPLATE allows the format of session names to be configured. Window names have the syntax:

```
#AAAnnn
```

must appear as the first char-

AAA alphabetic prefix, 1 to 4 letters.

nnn numeric suffix, 2 to 5 decimal digits.

Total must be 4 to 8 characters including "#".

Examples:

#TERM000 increments to #TERM999, then back to #TERM000. 1,000 unique names.

#P77 increments to #P99 then back to #P00. Shortest possible name. 100 unique names.

#AB12345 cycles to #AB99999 then back to #AB00000. 100,000 unique names (maximum allowed).

If GWN^TEMPLATE is not used, or does not follow the above rules, a default of #ZWN0001 is used, which is compatible with STN B19 and earlier.

GWN^TEMPLATE defines both the format of the name and the starting window name. As sessions are started, the numeric suffix is incremented until it reaches all nines, then the next window name wraps back to all zeroes. Using a short numeric suffix makes typing window names easier. Using a longer numeric suffix allows for more sessions before a window name is reused.

GWN^TEMPLATE may be used with or without GWN^FILE.

PARAM GWN^INITIAL RANDOM

If this param is present and is set to the value RANDOM, the initial value is randomly computed from the microsecond clock. Otherwise, the number in GWN^TEMPLATE, if present, is used, or else the default of 0001.

GWN^INITIAL may be used with or without GWN^FILE.

PARAM GWN^FILE <filename>

GWN^FILE names a central disc file where the next window name is stored. Normally, all STN processes would share the same file by using the same PARAM GWN^FILE value.

<filename> must name a disc file.

If the file does not exist, it is created as an unstructured disc file, code 1107, and initialized using GWN^TEMPLATE and GWN^INITIAL. If it cannot be created or written, the default of #ZWN0001 is used.

If the file exists, it is validated as containing a valid GWN record. If the GWN record is valid, STN allocates an initial block of window names as described below. The window name stored in the file overrides any GWN^TEMPLATE.

If the file exists but an error occurs while opening or reading the file, or the file does not contain valid GWN data, STN closes the file, generates an EMS warning and runs without GWN^FILE for the duration of the STN process. No recovery is attempted. If it cannot be created or written, the default of #ZWN0001 is used.

If <filename> is OFF, or the PARAM is omitted, then the default of #ZWN0001 is used.

PARAM GWN^BLOCKSIZE <nnn>

When GWN^FILE is used, GWN operates by allocating a block of consecutively numbered window names at a time. This allows multiple STN processes to use the same range of window names without duplicating any names. It also allows a restarted STN process to avoid duplicating names previously used.

GWN^BLOCKSIZE specifies the number of window names to be allocated in each block, in the range 10-1000. If GWN^BLOCKSIZE is not specified, or contains an illegal value, a default or 25 is used.

Allocation works as follows:

1. STN reads GWN file (with locking) to get the next window name.
2. This window name and the next <blocksize>-1 consecutive window names are reserved for use by this STN process.
3. STN adds <blocksize> to the numeric portion of the window name and rewrites (with unlock) GWN file.
4. STN then uses the reserved window names for new sessions. When the reserved list is exhausted, another allocation is performed.
5. If any error occurs reading or writing GWN^FILE, the file is closed and the default #ZWN0001 is used for the duration of the STN process.

GWN^BLOCKSIZE is automatically reduced if necessary so that is does not exceed a tenth of the numeric range defined by GWN^TEMPLATE. For example, with GWN^TEMPLATE #T00, there are only 100 names in the range, so the maximum is 10. For #PTY0000, the maximum is 1000.

With this allocation scheme, there may be some gaps in window numbering, but there will generally be no duplication, which can simplify tracking of windows.

GWN Related STNCOM Commands

INFO STN

Displays GWN parameters.

DYN_WIN_MAX

The existing DYN_WIN_MAX command is generally superseded by the features of GWN^TEMPLATE, but it is still allowed.

```
DYN_WIN_MAX nnn
```

nnn is the maximum number of window names, including zero (0). nnn must be in the range 100 to 100000, default 100000. DYN_WIN_MAX may be used to reduce the number of windows allowed by GWN^TEMPLATE. For example:

```
PARAM GWN^TEMPLATE #Z0000
STNCOM $STN ; DYN_WIN_MAX 250
```

cycles from #Z0000 to #Z0249, then back to #Z0000.

GWN [ALLOC]

STNCOM displays the GWN filename and details about the window name and option and optionally a new block of names.

This command always displays current information.

- GWN File name (or blank)
- Blocksize
- Next window name
- Last window name allocated (same as next if no GWN File)
- Maximum window number

If ALLOC is specified, a new block of session names is allocated from GWN^FILE. Since allocation is normally done automatically, ALLOC is intended for development use only. Any window names reserved by a previous GWN^FILE allocation but not yet used are discarded. The next session will begin with the number just allocated.

GWN Related EMS Events

EMS events are generated at GWN initialization, whenever allocations are made from GWN^FILE, and whenever any errors occur. Refer to the section on EMS events.

SCF and SPI

STN provides limited support for SCF and SPI:

- SCF may not be used to configure STN; all configuration and control is done using STNCOM.
- The subset of SPI commands used by NonStop™ ASAP is supported
- INFO / STATUS / STATS PROC.
- INFO / STATUS / STATS / LISTOPENS WINDOW <window>. Only single window may be specified. "*" for all windows is not supported.

  Starting with B08, SPI INFO WIN returns an additional token ZSTN^TKN^SSH^PROC 1005 (see ZSTNDDL) which contains the SSH process name for PTY sessions.
- STATS SERVICE <service>. Only single service may be specified. "*" for all services is not supported.
- NAMES SERVICE / WINDOW *
- NAMES (LISTOBJECTS) responses are limited to a single buffer with no error or continuation indication. SCF NAMES WINDOW $STN.* will return approximately 150-200 window names.
- Some fields have different interpretations.
- Some additional tokens are present. SCF and NonStop™ ASAP ignore these. See ZSTNDDL.

SPI support in STN is limited to the commands used for NonStop™ ASAP. These commands can also be used from SCF, but this is not recommended. STNCOM is required for all configuration and is recommended over SCF even for those commands which are supported from SCF.

EMS Events

The STN installation subvolume contains standard EMS files which provide additional details:

- ZSTNDDL DDL for event names
- ZSTNTMPL template output file for EMSDIST

It is recommended that ZSTNTMPL be installed using standard procedures.

Note: In the following event descriptions, event name and number are given, followed by the EMS template for this event. All references to <1> refer to the STN process that issued the event.

zstn-evt-stnlog value is 1003

"<1> STNLOG <2>"

<2> text

- CAUSE: STNLOG messages can be generated by other components and also by the STNCOM command STNLOG. The text is described in the documentation for the component which generated the message.
- EFFECT: Refer to other documentation.
- RECOVERY: Refer to other documentation.

zstn-evt-application-loop value is 1018

"<1> STN Application <2> is looping on window <3>"

<2> name of application

<3> STN window name

- CAUSE: An application has repeatedly attempted to perform output to a terminated session. (See STNCOM command REPLY_MAX_DELAY). The application process name and STN window name are displayed. This message is displayed once per session.
- EFFECT: None.
- RECOVERY: Review the application for proper error handling.

zstn-evt-auditcoll-start value is 1020

"<1> AUDITCOLL started to collector <2> version <3>"

<2> name of AUDITCOLL collector

<3> STN version and release date

- CAUSE: STNCOM command AUDITCOLL was used to open an EMS collector. This event is written to the specified collector, not to the standard $0 EMS event collector.
- EFFECT: Audit-type events will be written to the specified collector.
- RECOVERY: None; informational only.

zstn-evt-auditcoll-stop value is 1021

"<1> AUDITCOLL stopped"

- CAUSE: STNCOM command AUDITCOLL OFF was used. This event is written to the specified AUDITCOLL collector, not to the standard $0 EMS event collector.
- EFFECT: Events are no longer written to the audit collector. Normal EMS event processing to $0 continues.
- RECOVERY: None; informational only.

zstn-evt-auditcoll-sslmiscerr value is 1022

"<1> AUDITCOLL sslmiscerr <2> <3> <4> <5>"

<2>, <3>, <4> zero. Used only for SecurTN where this event has an alternate meaning.

<5> text from AUDITMSG.

- CAUSE: Generated when STNCOM command AUDITMSG is used. This event is written to the specified AUDITCOLL collector, not to the standard $0 EMS event collector.
- EFFECT: None.
- RECOVERY: None; informational only.

zstn-evt-auditcoll-service value is 1023

"<1> AUDITCOLL <2> <3> <4> service <5> Outcome <6>"

<2> full name of the window (\node.$stn.#window).

<3> remote IP address

<4> remote IP port

<5> window name only (#win)

<6> text "Granted" for a dedicated window, and "Granted" or "Denied" for a service.

- CAUSE: Generated on a session connection attempt to a service or dedicated window. Outcome is GRANTED or DENIED for a service, GRANTED for a dedicated window. This event is written to the specified AUDITCOLL collector, not to the standard $0 EMS event collector.
- EFFECT: None.
- RECOVERY: None; informational only.

zstn-evt-auditcoll-connect value is 1024

"<1> AUDITCOLL connect <2> <3> <4> <5> Client Info <6>"

<2> full name of the window (\node.#stn.#window)

<3> remote IP address

<4> remote IP port

<5> text "PLAIN" for unencrypted sessions, or "SECURE".

<6> encryption method.

- CAUSE: Generated when a new session is accepted from a remote workstation. The session can be either SECURE or PLAIN. This event is written to the specified AUDITCOLL collector, not to the standard $0 EMS event collector.

- EFFECT: None.
- RECOVERY: None; informational only.

zstn-evt-auditcoll-disconnect value is 1025

"<1> AUDITCOLL disconnect <2> <3> <4>"

<2> full name of the window (\node.#stn.#window).

<3> remote IP address.

<4> remote IP port.

- CAUSE: A session has terminated. This event is written to the specified AUDITCOLL collector, not to the standard $0 EMS event collector.
- EFFECT: None.
- RECOVERY: None; informational only.

zstn-evt-auditcoll-wsinfo value is 1026

"<1> AUDITCOLL <2> <3> <4> wsinfo <5> Outcome <6>"

<2> full name of the window (\node.#stn.#window),

<3> remote IP address.

<4> remote IP port.

<5> WSINFO text received from the workstation, if any.

<6> text "GRANTED" or "DENIED".

- CAUSE: WSINFO is set to REQUIRED or MATCH for a 6530 session. The information returned by the workstation is given, and the outcome is GRANTED if the session was allowed to continue or DENIED if the WSINFO requirements were not met. This event is written to the specified AUDITCOLL collector, not to the standard $0 EMS event collector.
- EFFECT: None.
- RECOVERY: None; informational only.

zstn-evt-max-outq value is 1027

"<1> STN window <2> exceeds max_outq <3>"

<2> name of window

<3> maximum number of queued output messages

- CAUSE: The number of queued output messages for a session exceeded the limit given by STNCOM command MAX_OUTQ. This is unusual application behavior.
- EFFECT: The session is terminated.
- RECOVERY: If the problem persists, contact Support.

zstn-evt-stop-process value is 1028

"<1> STN window <2> stopping process <3> status <4>"

<2> name of window

<3> process name

<4> status code

- CAUSE: STN is automatically stopping the process previously created for a dynamic window at session termination when KILL_DYNAMIC=Y.
- EFFECT: The specified process is stopped.
- RECOVERY: None; informational only.

zstn-evt-pool-used value is 1033

"<1> STN Buffer pool used <2> <3>% , used=<4>kw size=<5>kw"

Indicates STN memory pool usage goes above 80%, or back down below 80%.

<2> "OVER" or "UNDER"

<3> the threshold percentage as sent by the POOL_WARNING command (default 80)

<4> the current amount of memory used (unit=1024 words)

<5> the total size of the pool (unit=1024 words) as configured by PARAM POOL_SIZE

- CAUSE: Every minute STN checks the buffer pool usage and compares the percentage used against POOL_WARNING. If the amount has changed from under the threshold to over, or from over to under, this event is generated. This event also occurs one minute after startup time.
- EFFECT: If pool usage is UNDER, some sessions may terminate.
- RECOVERY: Use the POOL command to monitor pool usage. Increase PARAM POOL_SIZE and restart STN when convenient.

zstn-evt-th-open-err value is 1034

"<1> Open TH <2> error <3>"

<2> - Terminal Handler process name

<3> - Guardian open file error code

- CAUSE: I/O error opening the OSS Terminal Helper ($ZTTnn) process.
- EFFECT: The affected terminal session may hang.
- RECOVERY: None. Recovery is automatic. If other symptoms are noted, such as hanging sessions, include this EMS event when reporting the problem.

zstn-evt-th-writeread-err value is 1035

"<1> Writeread TH <2> error <3>"

<2> - Terminal Handler process name

<3> - Guardian writeread file error code

- CAUSE: I/O error writing to the OSS Terminal Helper ($ZTTnn) process.

- EFFECT: The affected terminal session may hang.
- RECOVERY: None. Recovery is automatic. If other symptoms are noted, such as hanging sessions, include this EMS event when reporting the problem. Recovery is automatic.

zstn-evt-gwn-file-err value is 1058

"<1> GWN File <2> error <3> on <4>"

<2> - GWN file name

<3> - Guardian file error code

<4> - File operation where error occured

- CAUSE: An error occured on the GWN file.
- EFFECT: STN will attempt to recover. Additional related EMS event(s) will give further information.
- RECOVERY: None, but see additional EMS events.

zstn-evt-gwn-file-created value is 1059

"<1> GWN File <2> Created"

<2> - GWN file name

- CAUSE: STN created a new GWN file based in GWN^FILE because the file did not already exist.
- EFFECT: GWN startup continues.
- RECOVERY: None; informational.

zstn-evt-gwn-file-init value is 1060

"<1> GWN File <2> Initialized to <3>"

<2> - GWN file name

<3> - Window name

- CAUSE: STN created a new GWN file
- EFFECT: The GWN file is initialized to the specified window name.
- RECOVERY: None; informational.

zstn-evt-gwn-file-bad-data value is 1061

"<1> GWN File <2> contains bad data <3>"

<2> - GWN file name

<3> - Sample of bad data

- CAUSE: STN encountered unexpected data in the GWN file.
- EFFECT: GWN is disabled.
- RECOVERY: Correct the problem with the file, purge the file, or change PARAM GWN^FILE to the proper filename, then restart STN.

zstn-evt-gwn-disabled value is 1062

"<1> GWN File disabled - using <2> session/window names"

<2> - Number of session/window names

- CAUSE: STN encountered an error with GWN processing as detailed in a previous event. This event also occurs once at STN startup, when no PARAM GWN^FILE is present.
- EFFECT: Future window names for this STN process use the traditional #ZWNnnnn scheme. If this error occurs for multiple STN processes, then duplicate #ZWN names can occur.
- RECOVERY: Correct the underlying error and restart the STN process.

zstn-evt-gwn-allocated value is 1063

"<1> GWN File <2> Allocated names <3> to <4>"

<2> - GWN file name

<3> - first window name allocated to this STN process

<4> - last window name allocated

- CAUSE: This STN process allocated (reserved) a block of window names from the GWN file.
- EFFECT: The specified window names will be used for future sessions for this STN process
- RECOVERY: None; informational.

zstn-evt-abend value is 1

"<1> Process abend due to <2>"

<2> provides a brief textual description

- CAUSE: An unrecoverable internal error was detected.
- EFFECT: The STN process will abend and usually create a ZZSA dump file. If a backup process is running, it will take over; if not, STN will terminate.
- RECOVERY: If STN is not running with a backup process, STN must be restarted. Forward the ZZSA file to Support.

zstn-evt-alloc value is 2

"<1> Allocatesegment err <2> POOL^SIZE <3> words"

<2> error code

<3> requested size in words

- CAUSE: An extended segment could not be allocated for the STN internal buffer pool.
- EFFECT: The STN process will abend and usually create a ZZSA dump file. STN will terminate and will not perform a backup takeover.
- RECOVERY: If PARAM POOL^SIZE is too large, and the disk volume containing the STN object file is full or fragmented, try freeing up some disk space, or carefully reduce the PARAM POOL^SIZE, then restart STN. If the problem persists, contact Support.

zstn-evt-starting value is 3

"<1> <2> program starting <3>"

<2> program name and version information

<3> additional copyright information

- CAUSE: The STN process has started.
- EFFECT: None.
- RECOVERY: None; informational only.

zstn-evt-param-error value is 4

"<1> Error in PARAM <2> <3>"

<2> parameter name

<3> value

- CAUSE: During STN startup, an error was found.
- EFFECT: The param is ignored, and STN startup proceeds without the param. Depending on the param, STN may not operate properly.
- RECOVERY: If the parameter is important, correct the error, then stop and restart STN.

zstn-evt-gftcom-start-err value is 5

"<1> Error <2> <3> starting GFTCOM^OBJECT <4>"

<2> error code

<3> detail error

<4> program name

- CAUSE: PARAM GFTCOM^OBJECT was specified but an error was encountered when trying to start the program indicated.
- EFFECT: The param is ignored, and STN startup proceeds without the parameter. Since this command is generally used for essential configuration commands, STN will probably not operate properly.
- RECOVERY: Correct the error, then stop and restart STN, or use STNCOM command to directly enter any required configuration commands.

zstn-evt-backup-started value is 6

"<1> Backup created in cpu <2>"

<2> cpu number

- CAUSE: STN created a backup process (a) after startup time when PARAM BACKUP is used, (b) after STNCOM BACKUPCPU command, (c) after a takeover, or (d) after a backup CPU became available.
- EFFECT: STN is now operating with a backup process.
- RECOVERY: None; informational only.

zstn-evt-backup-stopped value is 7

"<1> Backup stopped"

- CAUSE: The STN backup process stopped. Another EMS event may give additional information.
- EFFECT: STN runs without a backup. In some cases, STN will automatically restart the backup process immediately or after a backup CPU becomes available.
- RECOVERY: If backup operation is required, make the backup CPU available or use the STNCOM command BACKUPCPU to select another backup CPU.

zstn-evt-backup-start-err value is 8

"<1> Backup create error <2> <3>"

<2> error code

<3> detail error

- CAUSE: STN could not create a backup process due to a process_create_ error.
- EFFECT: STN runs without a backup. In some cases, STN will automatically restart the backup process immediately or after a backup CPU becomes available.
- RECOVERY: If backup operation is required, make the backup CPU available or use the STNCOM command BACKUPCPU to select another backup CPU.

zstn-evt-misc value is 9

"<1> <2>"

<2> Text. There are several variations of this event; currently, only one is listed.

"STN requires SAFECOM ADD DISKFILE <file-name>,PRIV-LOGON ON but it is <error-text> "

<file-name> - the STN object file name

<error-text> - could be:

Safeguard not running

Not configured

DISKFILE record not found

DISKFILE PRIV_LOGON OFF

- CAUSE: STN object not properly configured under Safeguard.
- EFFECT: STN cannot start dynamic service applications when SERVICE USER or LOGON is used.
- RECOVERY: Start Safeguard, then perform the following Safecom command for the STN object file:

```
ADD DISKFILE <stn-object-filename>,
PRIV-LOGON ON
```

This command can be performed when STN is running and takes effect immediately.

zstn-evt-checkalloc value is 10

"<1> Checkallocatesegment err <2>"

<2> error code

- CAUSE: STN could not allocate its internal buffer pool in the backup process due to an error condition.
- EFFECT: STN runs without a backup. STN will automatically restart the backup process.
- RECOVERY: If backup operation is required, use the STNCOM command BACKUPCPU to select another backup CPU.

zstn-evt-backup-loop value is 11

"<1> Backup creation loop - BACKUPCPU NONE assumed"

- CAUSE: The backup process repeatedly failed. Other EMS events will give additional information.
- EFFECT: STN runs without a backup until a STNCOM command BACKUPCPU is entered.
- RECOVERY: Correct the problem causing the backup failures, then use the STNCOM BACKUPCPU command.

zstn-evt-ckpt-fe value is 12

"<1> Backup checkpoint16file err <2>"

<2> error code

- CAUSE: Unable to communicate with backup process due to an error condition.
- EFFECT: The backup is stopped. STN will automatically restart the backup process.
- RECOVERY: None; informational only.

zstn-evt-ckopen-err value is 13

"<1> Checkopen err <2> file <3>"

<2> error code

<3> file name

- CAUSE: An error occurred during backup checkopen of a file.
- EFFECT: The backup is stopped. STN will automatically restart the backup process.
- RECOVERY: None; informational only.

zstn-evt-trace-start value is 14

"<1> Trace started to file <2> size <3>"

<2> trace file name

<3> size of the trace file

- CAUSE: An STN trace was started.
- EFFECT: None.
- RECOVERY: None; informational only.

zstn-evt-trace-stop value is 15

"<1> Trace stopped"

- CAUSE: An STN trace was stopped.
- EFFECT: None.
- RECOVERY: The binary trace file may now be forwarded to Support, or may be formatted using the GTRED program.

zstn-evt-trace-segment value is 16

"<1> Trace not started to <2> size <3> allocatesegment error <4>"

<2> extended segment file name

<3> size of the file

<4> error code

- CAUSE: An error was encountered when allocating an extended segment file.
- EFFECT: Tracing is not enabled.
- RECOVERY: Correct any errors in the trace filename, or select a disk with more available space, then retry the TRACE command.

zstn-evt-takeover value is 18

"<1> Backup process takeover due to: <2>"

<2> reason of the takeover, such as primary cpu failure, etc.

- CAUSE: STN backup process takeover.
- EFFECT: Backup process resumes STN operation. Any sessions active in the previous primary process are lost. New sessions will be accepted immediately. Depending on backup CPU availability, a new backup process is automatically started.
- RECOVERY: If the reason for the backup takeover, such as primary CPU failure, is understood, then no action is required. Otherwise, contact Support.

zstn-evt-trace-error value is 19

"<1> Trace not started to <2> size <3> error <4> / <5>"

<2> trace file name

<3> trace file size

<4> error code

<5> detail error

- CAUSE: An unusual error was encountered while opening a trace file.
- EFFECT: Tracing is not enabled.
- RECOVERY: Contact Support. Retry the TRACE command.

zstn-evt-trace-size-file value is 20

"<1> PARAM TRACE^SIZE must precede PARAM TRACE^FILE"

- CAUSE: PARAM TRACE^SIZE followed PARAM TRACE^FILE
- EFFECT: PARAM TRACE^SIZE is ignored, so the trace file is opened with the default size.
- RECOVERY: Reorder the PARAM list. STNCOM commands can be used to stop and restart the trace using the desired size without shutting down STN.

zstn-evt-reply-error value is 21

"<1> Reply error <2>"

<2> error code

- CAUSE: An unexpected file system error was returned by REPLYX.
- EFFECT: Usually none, unless other errors are noted.
- RECOVERY: If the problem persists, contact Support.

zstn-evt-stopping value is 22

"<1> Process stopping - SHUTDOWN command"

- CAUSE: The STNCOM command SHUTDOWN was entered.
- EFFECT: The STN process terminates. The backup process, if any, is stopped first. Any active sessions are immediately terminated.
- RECOVERY: Restart STN.

zstn-evt-cpuswitch value is 23

"<1> Primary process stopping - CPUSWITCH command"

- CAUSE: The STNCOM command CPUSWITCH was entered.
- EFFECT: A backup takeover occurs, and the old primary becomes the new backup.
- RECOVERY: None; informational only.

zstn-evt-enter-debug value is 24

"<1> Process entering debug"

- CAUSE: The STNCOM command DEBUG was entered.
- EFFECT: The STN process enters inspect/debug at its current home terminal. This will suspend all STN operation and can timeout any active sessions if the debug state is not exited within a short time.
- RECOVERY: The DEBUG command is generally used only by development and support staff.

zstn-evt-exit-debug *value is 25*

"<1> Process exiting debug"

- CAUSE: An inspect session from a previous DEBUG command finished.
- EFFECT: STN operation continues. Active sessions may timeout if the time spent in inspect mode was too long.
- RECOVERY: None; informational only.

Client Messages at the Remote Workstation

When a TN6530 client (terminal emulator) such as Win6530 or J6530 first connects to STN, several messages are displayed as the session is initiated. Each message begins with the letters "STN" followed by a two-digit message number for ease of identification.

STN00 Connected to STN version <version> <date/time> <window-name>

This is the first message displayed which confirms connection to STN (as distinct from Telserv or other Telnet servers). The STN version string is included. <window-name> is in the form \node.$process.#window.

STN01 Host IP <h> <subnet> Port <p> <window-name>

This is the second message, which confirms the NSK host IP address <h>, the TCP process name <subnet>, port number <p> and finally the full filename of the STN window in the form \node.$process.#window. This information can be useful for support purposes.

STN02 Services:

This message precedes the list of services displayed.

STN03 Terminal type <ttype> is not supported

The TN6530 client (terminal emulator) sent a terminal type identifier unknown to STN. Verify that the terminal emulator is properly set for TN6530 emulation.

STN04 Connected to Dedicated Window <window>

This message indicates that the session has been automatically connected to a dedicated window named <window> whose IP address matches the remote workstation.

STN05 Dedicated Window(s) are configured for this workstation IP address, but are already in use or otherwise unavailable. Session terminated.

Self-explanatory.

STN06

Reserved for future use.

STN07 SU Window not found

User entered #WINDOW name in response to the menu, but the specified window is not configured.

STN08 Window is not Type SU

User entered #WINDOW name in response to the menu, but the specified window is not configured as type SU.

STN09 Window is stopped by system operator

User entered #WINDOW name in response to the menu, but the specified window was stopped by STNCOM STOP/ABORT WINDOW command.

STN10 Connected to SU Window

User entered #WINDOW name in response to the menu, and the session was successfully connected to the requested window.

STN11 Service not found

User entered a service name in response to the menu, but the specified service is not configured.

STN12 Service is stopped by system operator

User entered a service name in response to the menu, but the specified service was stopped by STNCOM STOP/ABORT SERVICE command.

STN13 No Static Window available for this Service

User entered a service name in response to the menu, but the specified static service either has no windows configured, or all configured windows are in use or STOPPED.

STN14 Connected to Static Window <window>

User entered a service name in response to the menu, and the session was successfully connected to <window>, which was configured for the requested static service.

STN15 The Dynamic Service selected required a userid and password

STN15 Enter group.user:

For services with LOGON=REQ. Enter the Guardian userid or alias without the password.

STN16 Enter password:

This prompt follows the response to STN15.

STN17 Input error; proper syntax is group.user

Improper response to STN15 prompt.

STN18 Unknown userid or incorrect password; please wait ...

This follows the response to the STN16 prompt. After a delay to discourage hackers and automated logon attacks, the STN15 prompt is repeated. After three STN18 consecutive logon failures, the session is terminated.

STN19 Add Window failed for Dynamic Service

User entered a dynamic service name in response to the menu, but a new dynamic window could not be added, usually due to a resource shortage. Notify Support.

STN20 Starting Dynamic Service application

STN is starting the application for the requested dynamic service.

STN21 Dynamic Service Application Creation Error

STN was not able to start the application for the requested dynamic service. An additional message STN22-STN34 is displayed with error details from PROCESS_CREATE_.

STN22 file error <fe> on PROGRAM file

PROCESS_CREATE_ error 1: File system status <fe> on PROGRAM file.

STN23 file error <fe> on LIB file

PROCESS_CREATE_ error 3: File system status <fe> on LIB file.

STN24 file error <fe> on SWAP file

PROCESS_CREATE_ error 5 or 6: File system status <fe> on SWAP file.

STN25 file error <fe> on HOME TERM file

PROCESS_CREATE_ error 8 or 9: File system status <fe> on HOME file.

STN26 CPU(s) configured for this service are down

PROCESS_CREATE_ error 10: none of the CPUs for this service are running.

STN27 file error <fe> on process name

PROCESS_CREATE_ error 11: File system status <fe> on HOME file.

STN28 PROGRAM file format error <detail>

PROCESS_CREATE_ error 12: PROGRAM file error, see detail.

STN29 LIB file format error <detail>

PROCESS_CREATE_ error 13: LIB file error, see detail.

STN30 no pcb available

PROCESS_CREATE_ error 15: no pcbs available.

STN31 unlicensed privileged program

PROCESS_CREATE_ error 17.

STN32 library conflict

PROCESS_CREATE_ error 18.

STN33 PROG and LIB files the same

PROCESS_CREATE_ error 19.

STN34 process_create_ error <status> substatus <substatus>

PROCESS_CREATE_ error <status> with detail <substatus>.

STN35 **WARNING** Terminal will be disconnected if it stays idle...

When BANNER_TIMEOUT or INPUT_TIMEOUT is in effect and there has been no input (and no output if OUTPUT_RESET=Y), STN35 is displayed every minute when the inactive time period is within IDLE_WARNING minutes of the timeout.

STN36 Terminal was idle too long! Disconnecting...

When BANNER_TIMEOUT or INPUT_TIMEOUT is in effect and there has been no input (and no output if OUTPUT_RESET=Y), STN36 is displayed and 10 seconds later the session is terminated.

STN37 BLAST <text>

STNCOM command BLAST was used to force <text> to be sent to all sessions.

STN38 No application program active on this terminal for <n> seconds. Session terminated.

At the beginning of a session, OPENER_WAIT seconds have elapsed and no application has opened the window. See OPENER_WAIT for details.

STN39 Session terminated - application request (control 12) <time>

The application has disconnected the session via control 12. This is normal termination for some applications, like TACL logoff.

After session termination, 6530 terminals will always be left in conversational (ITI) mode, and the terminal display is erased.

STN39 Session terminated - application closed terminal <time>

The application has closed the window and AUTODEL_WAIT seconds have elapsed. This is normal termination for some applications, for instance TACL exit. See AUTODEL_WAIT for details.

After session termination, 6530 terminals will always be left in conversational (ITI) mode, and the terminal display is erased.

STN41 The requested dynamic service application was started, but did not connect to this window within 60 seconds. The application, and this session are being stopped.

This generally indicates a programming error in the application for the dynamic service. Contact the system administrator.

STN42 open (for startup message) error on process <p> fe <fe>

For dynamic windows, STN tried to open the newly created application process <p> to pass the startup message, but the open was rejected with file system error <fe>. Contact the system administrator.

STN43 write (for startup message) error on process <p> fe <fe>

For dynamic windows, STN opened the newly created application process <p> to pass the startup message, but the write was rejected with file system error <fe>. Contact the system administrator.

STN44 Application <$name> has connected to this window

STN has detected an open from the application program. The next message will be from the application (e.g. TACL prompt). <$name> is the application process name.

STN46 Secure SSH session: <SSH info>

This is an informational message to emphasize that the session is secure. Encryption details are provided.

STN48 <window-or-service>

This is an informational message to echo the response to the menu prompt. This is especially useful when the service name is automatically entered by the terminal emulator.

STN50 Negotiation timeout - check Line Mode setting in terminal emulator. Session terminated.

Telnet IAC negotiations did not complete within 20 seconds.

STN51 Workstation IP address not in range for requested service

The IP address of the remote workstation is not defined in the IPRANGE, or the IPRANGE is not defined.

STN54 session timed out waiting for user logon response

A session connected to a SERVICE with LOGON REQ, but the user did not respond to the logon prompt.

STN57 This 6530 emulator does not support required WSINFO

See STNCOM command WSINFO.

STN58 WSINFO address does not match network address

See STNCOM command WSINFO.

STN59 Input discarded

For an SSH session with no read active (TACL PAUSE-d etc), a very large amount of keyboard input was received. Further input is discarded.

STN70 No existing window available for resilient service, window <win> added

A resilient service was requested, but no previously created windows were available. STN creates a new window and starts the application.

STN70 Reconnecting to resilient window <win> Last access: <date> <time>

Connection to a resilient service where an existing window from a previous session has been reconnected to the current session.

STN70 application <$pname | pid | cpu,pin> <program-filename>

When reconnecting to a resilient window, one line is displayed (up to 12 lines) for each process which had the window open. For Guardian processes, the program object file name and $pname or cpu,pin is edited; for Posix processes, the pid is displayed in hex.

STN70 Additional openers not listed

When reconnecting to a resilient window, one line is displayed (up to 12 lines) for each process which had the window open. For Guardian processes, the program object file name and $pname or cpu,pin is listed; for Posix processes, the pid is displayed in hex. This message is displayed if there were more than 12 processes and the remainder had been discarded.

STN70 no application active on this window

When reconnecting to a resilient window, no application programs were open. The window is effectively unusable.

STN71 Userid not allowed for this service

The selected service included a USER parameter, and the userid entered at the keyboard (or automatically supplied) does not match. The session is terminated.

STN72 Using userid from SSH

SYSTEM-USER is being used instead of STN15/STN16 prompt.

STN73 Using SSH_Default_Svc

CI-PROGRAM *MENU* (without anything following *MENU*) and the service specified by SSH_DEFAULT_SVC is used.

STN74 Dynamic Service Session Limit Exceeded

The selected service included a LIMIT parameter and there are already <limit> sessions active. The session is terminated.

STN75 Service/window required by SSH user config not available

Service/window required by SSH configuration FORCE not available.

STN76 Authenticated <auth-mechanism> client: <client-display-name>

At session startup, this confirms the authentication mechanism and the user name.

STN81 Client IP address <n.n.n.n>port <nnn>

The TCP/IP address and port number of the remote client workstation, as reported by NonStop TCP/IP socketlib.

STN82 SSH external user <ext-user>, Guardian system user <group.user>

The user names reported by SSH.

STN83 WSINFO User <user> IPaddr <n.n.n.n> Host <PC-hostname>

For sessions when WSINFO is set to QUERY or REQUIRED, the information reported by the client workstation 6530 emulator is displayed.

STN84 Cannot create new session - no dst available

For Type Dynamic and Pathway services, a dynamic window could not be created because the maximum number of dynamic windows DYN_WIN_MAX has been exceeded.

STN87 Too many services, <NN> additional services not displayed

STN02 only lists first 200 service names.

STN94 Userid <group.user> provided by SSH not valid

SSH sessions with *MENU* and an SSH Guardian system user in group.user format that do not match SERVICE USER are now terminated with this message.

STN94 Userid <alias> provided by SSH not valid

SSH sessions with *MENU* and an SSH Guardian system user in alias format that matches SERVICE USER, but the STN object does not have PRIV-LOGON set via the command:

```
Safecom ADD DISKFILE STN,
PRIV-LOGON ON
```

STN Application I/O Handling

Standard SETMODE Functions:

6	line spacing
7	automatic LF
8	block mode / conversational mode
9	interrupt character definitions
11	break owner
12	break mode
14	interrupt character enable/disable
20	echo
22	set /retrieve baud rate. Only used to retrieve values detected by setmode 204
23	character size (always in 8 bit mode)
28	initialize all setmodes to default values except block mode, then apply any SCRIPT associated with the window
144	set ignored; retrieve always returns hex 8200 0900
258	full duplex

Extended SETMODE Functions (unique to STN):

201	Only used with special terminals. Enable timing mark flow control. P1=0 (default) disables the feature. 0<P1<10000 specifies the number of bytes to send before sending IAC DO TM and waiting for a response. P2 is a timeout in seconds (range 1-3600, default 3600); if no response is received to IAC DO TM, output proceeds after the timeout.
202	Only used with special terminals. Enable baud rate detection from remote client using rfc 1079. Default P1=0 disables, P1>0 enables. P2 presently unused. The baud rate detected can be retrieved by setmode 204 as a 32-bit integer or by setmode 22 which maps selected baud rates 75-19200 to values 1-15 (using the traditional ATP coding for setmode 22) and other baud rates to 0.
203	Only used with special terminals. P1=0 default compatible with previous releases. P1=1 discard any data after an application read is satisfied due to maximum read count up to and including the next line end (ascii CR). P2 presently unused.
204	Only used with special terminals. Retrieves the speed detected by setmode 202. P1 is the high order word, P2 is the low order word. Setting this value affects only the value returned in future setmode 204 calls.

205	Only used with special terminals. p1=1 disables echo of ascii EOT (hex 04). p1=0 (default) is compatible with previous releases and handles EOT like other characters for echo purposes
206	Only used with special terminals. P1=1 disables interrupt character handling for ascii BS/CTRL-H (hex 06), ascii CAN/CTRL-X (hex 18), and EM/CTRL-Y (hex 19), and also the 6530 control character ascii ENQ (hex 05), p1=0 (default) is compatible with previous releases
207	P1 and P2 are ignored; ascii "ST" and "N0" are returned as last parameters. This can be used by applications to verify that the file is really an STN process. Telserv will never respond with this string.
208	P1=1 (default) When a Posix read is active, signal characters (like control-C) generate Guardian Break when break is enabled. P1=0 generates the Posix signal.
212	To control setting of Pending^140 flag on session termination. Default is 1 which sets Pending^140 on session termination. 212,0 means Pending^140 flag is never set. Pending^140 controls the response to application I/O requests when no session is active. Pending^140 set: Control 11 clears pending^140 and waits for a new session. Control 12 is ignored. All other requests are rejected with FESESSDOWN. Pending^140 clear : Setmodes are handled normally, but any changes may be re-initialized when a new session starts. All other requests are handled the same as above for Pending^140 set.
214	Used to override results of deviceinfo (and related calls) against a window. Open a window and use setmode 214 with both P1 and P2 specified. If P1 is nonzero, then it overrides the device type and device subtype returned by deviceinfo. The device type is taken from P1.<4:9> and the subtype from P1.<10:15>. If P2 is nonzero, then it overrides the record length returned by deviceinfo. No range checking is done on either parameter. Setmode 214 P1 and P2 both default to zero when a window is added, and the value is not changed or reset by session termination or startup (unless part of a SCRIPT). Setmode 214 may be used with ADD SCRIPT, but with a static window the script will not be applied until the first session connects.

Standard SETPARAM Functions:

37	break handling

Extended SETPARAM Functions (unique to STN):

200	returns STN vproc information, example: Gemini STN A50 22JUN2006
201	returns the IP address of the remote workstation as reported by NonStop TCP/IP call accept_nw (4 bytes)
202	returns the WSINFO host name or empty string
203	returns the WSINFO ip address or empty string
204	returns the WSINFO user name or empty string
205	returns the IP port number of the remote workstation as reported by NonStop TCP/IP call accept_nw (2 bytes)
206	returns the IP address of the NonStop host as reported by NonStop TCP/IP call getsockname (4 bytes)
207	returns the IP port of the NonStop host as reported by NonStop TCP/IP call getsockname (2 bytes)
208	returns the Kerberos Principal Name if available for PTY sessions.
209	Info from WSINFO domain or empty string.
210	Info from WSINFO netbios or empty string.
211	Info from WSINFO client or empty string

Monitoring and Auditing

Introduction

The SSH2 process writes two kinds of messages that allow users to analyze its operation:

- Log messages are intended to show the overall functioning of such processes as startup, normal operation, and error conditions. Log messages can be written to a file, to a console device, or an event collector process.
- Audit messages are intended to provide a view of operations executed from an auditor's perspective. Therefore audit messages only deal with specific events on specific objects with specific outcomes. Audit messages can be written to a file or to a console device.

This chapter will describe the configuration and interpretation of both kinds of messages.

Additionally the status of the SSH2 process, of sessions, channels and openers can be helpful for monitoring the operation of the SSH2 process (see STATUS commands in chapter "SSHCOM Command Reference").

Log Messages

Content of Log Messages

SSH2 writes log messages either to a terminal or to a file. The following example shows the log messages it creates during startup:

```
$US SSH87A 20> RUN SSH2/name $SSH42/ ALL;SUBNET $ZTC1;PORT 42022;PTYSERVER $SSH42
$SSH42|09Dec09 20:00:17.54|20|-----------------------------------------------------
--------
$SSH42|09Dec09 20:00:17.54|10|comForte SSH2 version
T9999H06_01Dec2009_comForte_SSH2_0087
$SSH42|09Dec09 20:00:17.55|10|config file:  '(none)'
$SSH42|09Dec09 20:00:17.56|20|object filename is  '\NPNS01.$US.SSH87A.SSH2'
$SSH42|09Dec09 20:00:17.56|20|object subvolume is '\NPNS01.$US.SSH87A', priority is
168
...
$SSH42|09Dec09 20:00:17.69|20|dumping configuration:
...
$SSH42|09Dec09 20:00:17.76|10|SSH config database SSHCTL opened.
$SSH42|09Dec09 20:00:17.77|20|parameter SUBNET was evaluated
$SSH42|09Dec09 20:00:17.77|20|DEFINE =TCPIP^PROCESS^NAME was set to <\NPNS01.$ZTC1>
$SSH42|09Dec09 20:00:17.77|20|TCP/IP process is $ZTC1
$SSH42|09Dec09 20:00:17.84|20|DEFINE =SSH2^PROCESS^NAME was set to <\NPNS01.$SSH42>
$SSH42|09Dec09 20:00:17.84|10|Initializing SSH2 ADMIN run mode.
$SSH42|09Dec09 20:00:17.84|10|Initializing SSH2 CLIENT run mode.
$SSH42|09Dec09 20:00:17.84|10|Initializing SSH2 DAEMON run mode.
$SSH42|09Dec09 20:00:18.04|10|Loading private key from HOSTKEY
$SSH42|09Dec09 20:00:18.23|30|Host key algorithm: ssh-dss
```

```
$SSH42|09Dec09 20:00:18.23|30|Host key MD5 fingerprint:
b0:c7:86:e6:63:b8:2d:4b:b7:78:84:ec:dc:33:ed:c9
$SSH42|09Dec09 20:00:18.23|30|Host key Bubble-Babble: xetig-fegyg-pidyn-babyl-kefod-
sigeh-danyb-gykyl-sebuc-curul-fuxyx
$SSH42|09Dec09 20:00:18.23|10|SSH2 Server listening on interface 0.0.0.0, port 42022
```

The following example shows some log messages when an SFTP client connects, issues some commands, and disconnects:

```
$SSH42|09Dec09 20:15:42.96|50|10.0.0.78:3133: accepted connection from client
$SSH42|09Dec09 20:15:42.98|50|10.0.0.78:3133: client version string: SSH-2.0-
OpenSSH_3.8.1p1
$SSH42|09Dec09 20:15:43.05|40|10.0.0.78:3133: SSH session established.
$SSH42|09Dec09 20:15:43.07|20|10.0.0.78:3133: none authentication for user 'comf.us'
not allowed
$SSH42|09Dec09 20:15:43.15|40|10.0.0.78:3133: signature ok, authentication of comf.us
successful
$SSH42|09Dec09 20:15:43.17|50|10.0.0.78:3133: channel request for subsystem sftp,
launching sftp server
$SSH42|09Dec09 20:15:43.25|50|10.0.0.78:3133: launched program
\NPNS01.$US.SSH87A.SFTPSERV successfully (\NPNS01.$Z2QB:45580213)
$SSH42|09Dec09 20:17:20.24|40|10.0.0.78:3133: SSH session terminated
```

Incoming ssh connections are identified by the remote IP address and remote port, separated by a colon ("10.0.0.78:2928" in the above example). This log id is displayed as SESSION-LOG-ID in the output of SSHCOM command STATUS SESSION:

```
% status session *
status session *
SID   SESSION-LOG-ID        R USER-NAME            STRT-TIM      CHCNT AUTH-USR
1     10.0.0.78:3133        S COMF.US              09Dec09,20:15 1     comf.us
%
```

Using the WHERE option with the STATUS SESSION command the session status can be filtered to display just the status for a given session log id (while the session is still established):

```
% status session *, where session-log-id = "10.0.0.78:3133"
status session *, where session-log-id = "10.0.0.78:3133"
SID   SESSION-LOG-ID        R USER-NAME            STRT-TIM      CHCNT AUTH-USR
1     10.0.0.78:3133        S COMF.US              09Dec09,20:15 1     comf.us
%
```

Please see chapter "SSHCOM Command Reference" for details about the STATUS SESSION command.

Note: Since IPV6 address support, the session-log-id may become too large for display in the STATUS SESSION brief output. It has been removed in SPR T0801^ABE and can be determined via STATUS SESSION *, detail. Starting with SPR T0801^ABE, the brief output now contains the following columns: SID, R, USER-NAME, STRT-TIM (Start-time), CHCNT(Channel-count), AUTH-USR (Authenticated user), and AUTH (Authentication-method).

Log Level

Each log message has a "level" associated with it. The level is a number between 0 and 100 and is shown immediately after the timestamp. A lower number means a higher importance of the message. The parameters LOGLEVELFILE, LOGLEVELCONSOLE, and LOGLEVELEMS control which messages are generated for the various log destinations (also see next section): only log messages with a level greater than or equal than the level configured for the target will be generated. The log level configuration should be chosen as follows:

- 50 (default): log normal operation
- 30: only log startup messages and warnings
- 70: detailed diagnostic messages. Should only be set if the additional verbosity is really required.
- 100: very detailed diagnostic messages. This configuration is not recommended for production environments as it will create significant overhead.

Destinations for Log Messages

The SSH2 component can log to the following destinations:

- A file configured with the LOGFILE parameter.
- An process-internal memory cache for log message (parameters LOGLEVELCACHE, LOGCACHESIZE)
- A device configured with the LOGCONSOLE parameter.
- An event collector process configured with the LOGEMS parameter.

By default, the SSH2 component logs messages only to the home terminal. Logging to a file or EMS is not enabled by default. It is possible to log to multiple destinations. Which combination is best will depend on your operative environment. The following shows some examples on how to combine the log destinations in different scenarios:

- Getting used to SSH2, "experimenting": It may be easiest to start SSH2 with the default settings. In that case SSH2 will issue log messages to the home terminal only, making it easy to view the messages. Note that you cannot start the SSH2 component NOWAIT this way. It may be helpful to raise the LOGLEVEL to 100 in that case.

  ```
  LOGFILE *
  LOGEMS *
  LOGLEVELCONSOLE 100
  LOGCONSOLE %
  ```

- Log to EMS and only log startup and severe messages:

  ```
  LOGFILE *
  LOGCONSOLE *
  LOGEMS $0
  LOGLEVELEMS 30
  ```

- Log normal operations to a file and startup and severe messages to EMS:

  ```
  LOGCONSOLE *
  LOGFILE $vol.subvol.logfile
  LOGLEVELFILE 50
  LOGEMS $0
  LOGLEVELEMS 30
  ```

- Log normal operations to a file and startup and severe messages to EMS, log detail information to log cache and write content to the log file via SSHCOM command FLUSH LOGCACHE only after specific events:

  ```
  LOGCONSOLE *
  LOGFILE $vol.subvol.logfile
  LOGLEVELFILE 50
  LOGEMS $0
  LOGLEVELEMS 30
  LOGLEVELCACHE 85
  ```

Writing to the log cache causes the least overhead. If detailed log messages need to be analyzed, then it is often best to set the value of LOGLEVELCACHE to a higher value (e.g. via SSHCOM command SET LOGLEVELCACHE) and leave the parameter LOGLEVELFILE at the default level. After the event occurred that is of interest the messages in the log cache should then be written to the log file using SSHCOM command FLUSH LOGCACHE (see section "SSHCOM Command Reference"). The SSHCOM command ROLLOVER LOGFILE can be used to force the log file rollover allowing to keep the log file small.

For details about the parameters controlling the log behavior please refer to the LOG parameters in the chapter titled "Configuring and Running SSH2".

See the section on "Log File/Audit File Rollover", on how to look at the content of a log file.

Customizing the Log Format

SSH2 allows users to customize certain aspects of the appearance of log messages. Using the LOGFORMAT parameter, you can add the current date to the log message header. Please refer to the "LOGFORMAT" parameter description in the "SSH2 Parameter Reference" (chapter "Configuring and Running SSH2") for details.

Audit Messages

Content of Audit Messages

Audit messages are generated for various kinds of events:

- Authentication for a remote user.
- Starting of a SSH-subsystem such as SFTP.
- Opening of a file.
- Closing of a file.

Each audit message has a result: there can be a failure, or they can be granted or denied.

An individual audit message looks as follows:

$SSH49|22Dec10 15:20:47|10.0.0.78:1218: comf.us@10.0.0.78 authentication granted (method password): password ok. System user: COMF.US with the individual components as follows (from left to right):

- process name ("$SSH49")
- timestamp ("22Dec10 15:20:47")
- session identifier in SESSION-LOG-ID format ("10.0.0.78:1218"), if available
- local user id (present only in some audit messages)
- user and remote IP address ("comf.us@10.0.0.78")
- a string describing the operation and the outcome ("authentication granted (method password): password ok ")

Sample Audit Messages

The following listing shows the audit messages written for a single download of a file "/G/data1/ushome/test6" from the user "comf.us" at remote IP address 10.0.0.78:

```
$SSH49|22Dec10 15:31:12|10.0.0.78:1256: comf.us@10.0.0.78 authentication granted
(method password): password ok. System user: COMF.US
$SSH49|22Dec10 15:31:13|10.0.0.78:1256(COMF.US): comf.us@10.0.0.78 subsystem sftp
granted
$SSH49|22Dec10 15:31:13|10.0.0.78:1256(COMF.US): comf.us@10.0.0.78 list
/G/data1/ushome granted
$SSH49|22Dec10 15:31:22|10.0.0.78:1256(COMF.US): comf.us@10.0.0.78 open
/G/data1/ushome/test6 (mode read) granted (error 0)
$SSH49|22Dec10 15:31:25|10.0.0.78:1256(COMF.US): comf.us@10.0.0.78 close
/G/data1/ushome/test6: size 173, 173 bytes read, 0 bytes written
```

The following shows an audit message for a user trying to access the system with a non-existing username ("wronguser"):

```
$SSH49|22Dec10 15:43:07|172.16.123.103:1831: wronguser@172.16.123.103 authentication
failed (method none): System user 'wronguser' does not exist.
```

The following shows an audit message for a user trying to access the system with an existing user name, yet with an invalid public key:

```
$SSH49|23Dec10 15:57:23|172.16.123.110:3945: comf.us@172.16.123.110 terminated session

$SSH49|23Dec10 15:57:23|172.16.123.110:3945: comf.us@172.16.123.110 authentication
denied (method publickey): authentication aborted by client.
```

The following shows an audit message for a user trying to access the system with an existing user name that is frozen:

```
$SSH49|23Dec10 17:16:07|172.16.123.110:1708: comf.us@172.16.123.110 authentication
failed (method none): User is frozen.
```

The following shows an audit message for a user trying to access a file for which his SYSTEM-USER has no access rights:

```
$SSH49|23Dec10 17:22:42|172.16.123.110:1303(COMF.US): comf.us@172.16.123.110 open
/tmp/secret/file (mode read) failed (error 4013)
```

Destinations for Audit Messages

Similar as with log messages, the SSH2 component can send audit messages to three destinations:

- a file configured with the AUDITFILE parameter
- a device configured with the AUDITCONSOLE parameter
- a collector configured with the AUDITEMS parameter

By default, the SSH2 component does not write audit messages at all. It is possible to audit to one or more destinations at the same time.

Note that audit messages do not have a "level" as log messages have, auditing is either turned on to a destination or it is not.

See the section "Log File/Audit File Rollover" for information on how to assess the content of an audit file.

Customizing the Audit Format

SSH2 allows users to customize certain aspects of the appearance of audit messages. Using the AUDITFORMAT parameter, you can add the current date to the log message header. Please refer to the AUDITFORMAT parameter description for details.

Audit Reports

No tool is provided with SSH2 to create audit reports. However, given the simple format of the audit messages, any tool with sufficient text filtering capabilities can be used to create reports.

Using OSS to look at the audit file (see section "Viewing File Contents from OSS"), it is possible to create flexible reports with brief commands. If you need help in doing so, please contact the HP or comForte support team, depending on which product you are using.

List of Audit Messages

The following table shows the complete list of audit messages as created from release 89 on.

Note: Not all audit event variations (with different conditions) are currently used but may be in the future. Token values can be empty. Audit event pattern can change in the future.

Event Id	Event Name	Conditions	Pattern	Token Values
1	AuthenticationEvent	Authentication successful, method not publickey and not gssapi-with-mic	"%sessionId: %user@%remoteAddress %action %outcome (method %method): %reason. System user: %systemUser"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'authentication' %outcome: 'granted' %method: authentication method %reason: reason
		Authentication successful, method publickey or gssapi-with-mic	"%sessionId: %user@%remoteAddress %action %outcome (method %method, %publickeyOrPrincipal): %reason. System user: %systemUser"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'authentication' %outcome: 'denied' or 'failed' %method: authentication method % publickeyOrPrincipal: name of publickey or principal name %reason: reason
2	AuthenticationEvent	Authentication failed, Method not publickey and not gssapi-with-mic	"%sessionId: %user@%remoteAddress %action %outcome (method %method): %reason."	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'authentication' %outcome: 'granted' %method: authentication method %reason: reason
		Authentication failed, Method publickey or gssi-with-mic	"%sessionId: %user@%remoteAddress %action %outcome (method %method, %publickeyOrPrincipal): %reason."	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'authentication' %outcome: 'denied' or 'failed' %method: authentication method % publickeyOrPrincipal: name of publickey or principal %reason: reason
3	TerminateSessionEvent		"%sessionId: %user@%remoteAddress terminate session"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address
4	SubsystemEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'subsystem' %object: name of subsystem %outcome: 'granted'

Event Id	Event Name	Conditions	Pattern	Token Values
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘subsystem %object: name of subsystem %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘subsystem’ %object: name of subsystem %outcome: ‘denied’ or ‘failed’
5	SftpOpenFileEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome (mode %mode)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘open’ %object: file name %outcome: ‘granted’ %mode: file open mode (‘read’ or ‘write’)
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object (mode %mode) %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘open’ %object: file name %mode: file open mode (‘read’ or ‘write’) %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome (mode %mode)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘open’ %object: file name %outcome: ‘denied’ or ‘failed’ %mode: file open mode (‘read’ or ‘write’)
6	SftpTouchFileEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome (mode %mode)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘touch’ %object: file name %outcome: ‘granted’

Event Id	Event Name	Conditions	Pattern	Token Values
				%mode: file open mode (‘read’ if file exists or ‘write’ if file does not exist)
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object (mode %mode) %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘touch’ %object: file name %mode: file open mode (‘read’ if file exists or ‘write’ if file does not exist) %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome (mode %mode)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘touch’ %object: file name %outcome: ‘denied’ or ‘failed’ %mode: file open mode (‘read’ if file exists or ‘write’ if file does not exist)
7	SftpReadFile Event	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘read’ %object: file name %outcome: ‘granted’
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘read’ (remote error) or ‘write local file (local error)’ %object: file name %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘read’ (remote error) or ‘write local file (local error)’ %object: file name %outcome: ‘denied’ or ‘failed’

Event Id	Event Name	Conditions	Pattern	Token Values
8	SftpWriteFileEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'write' %object: file name %outcome: 'granted'
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'write' (remote error) or 'read local file (local error)' %object: file name %outcome: 'denied' or 'failed' %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'write' (remote error) or 'read local file (local error)' %object: file name %outcome: 'denied' or 'failed'
9	ftpCloseFileEvent	Successful	"%sessionId: %user@%remoteAddress %action %object: size %size, %bytes_read bytes read, %bytes_written bytes written"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'close' %object: file name %size: file size %bytes_read: number of bytes read %bytes_written: number of bytes written
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object (%error): size %size, %bytes_read bytes read, %bytes_written bytes written"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: 'close' %object: file name %error: error detail %size: file size %bytes_read: number of bytes read %bytes_written: number of bytes written
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object: size %size, %bytes_read bytes read, %bytes_written bytes written"	%sessionId: SESSION-LOG-ID %user: SSH username

Event Id	Event Name	Conditions	Pattern	Token Values
				%remoteAddress: remote IP address %action: ‘close’ %object: file name %size: file size %bytes_read: number of bytes read %bytes_written: number of bytes written
10	SftpPurgeFileEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘purge’ %object: file name %outcome: ‘granted’
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘purge’ %object: file name %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘purge’ %object: file name %outcome: ‘denied’ or ‘failed’
11	SftpRename Event	Successful	"%sessionId: %user@%remoteAddress %action %object to %newname %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘rename’ %object: old file name %newname: new file name %outcome: ‘granted’
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object to %newname %outcome (error %error)	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘rename’ %object: old file name %newname: new file name %outcome: ‘denied’ or ‘failed’ %error: error detail

Event Id	Event Name	Conditions	Pattern	Token Values
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object to %newname %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘rename’ %object: old file name %newname: new file name %outcome: ‘denied’ or ‘failed’
12	SftpListDirEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘list’ %object: directory name %outcome: ‘granted’
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘list’ %object: directory name %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘list’ %object: directory name %outcome: ‘denied’ or ‘failed’
13	SftpMkDirEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘mkdir’ %object: directory name %outcome: ‘granted’
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘mkdir’ %object: directory name %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address

Event Id	Event Name	Conditions	Pattern	Token Values
				%action: ‘mkdir’ %object: directory name %outcome: ‘denied’ or ‘failed’
14	SftpRmDirEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘rmdir’ %object: directory name %outcome: ‘granted’
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘rmdir’ %object: directory name %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘rmdir’ %object: directory name %outcome: ‘denied’ or ‘failed’
15	SftpSymlink Event	Successful	"%sessionId: %user@%remoteAddress %action %object target %link %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘symlink’ %object: file name %link: link name %outcome: ‘granted’
		Failed, error detail available	"%sessionId: %user@%remoteAddress %action %object target %link %outcome (error %error)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘symlink’ %object: file name %link: link name %outcome: ‘denied’ or ‘failed’ %error: error detail
		Failed, error detail not available	"%sessionId: %user@%remoteAddress %action %object target %link %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘symlink’ %object: file name %link: link name

Event Id	Event Name	Conditions	Pattern	Token Values
				%outcome: ‘denied’ or ‘failed’
16	PtyEvent	Successful	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘ptyallocate’ %object: pty name %outcome: ‘granted’
		Failed	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘ptyallocate’ %object: (empty) %outcome: ‘denied’ or ‘failed’
17	ShellEvent	No forced command	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘shell’ %object: shell program %outcome: ‘granted’, ‘denied’ or ‘failed’
		Forced command	"%sessionId: %user@%remoteAddress %action %object %outcome (forced command: %forcedcommand)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘shell’ %object: shell program %outcome: ‘granted’, ‘denied’ or ‘failed’ %forcedCommand: forced command
18	ExecEvent	No forced command	"%sessionId: %user@%remoteAddress %action %object %outcome"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘exec’ %object: shell program %outcome: ‘granted’, ‘denied’ or ‘failed’
		Forced command	"%sessionId: %sessionId: %user@%remoteAddress %action %object %outcome (forced command: %forcedcommand)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘exec’ %object: shell program

Event Id	Event Name	Conditions	Pattern	Token Values
				%outcome: ‘granted’, ‘denied’ or ‘failed’ %forcedCommand: forced command
19	ForwardEvent	Direct	"%sessionId: %user@%remoteAddress %action %object %outcome (%fromAddr:%fromPort->%toAddr:%toPort)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘forward’ %object: ‘direct-tcpip’ %outcome: ‘granted’ or ‘denied’ or ‘failed’ %fromAdd: from address %fromPort: from port %toAdd: to address %toPort: to port
		Not Direct	"%sessionId: %user@%remoteAddress %action %object %outcome (%fromAddr:%fromPort->remote, accepted on %toAddr:%toPort)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘forward’ %object: ‘forward-tcpip’ %outcome: ‘granted’ or ‘denied’ or ‘failed’ %fromAdd: from address %fromPort: from port %toAdd: to address %toPort: to port
19	ListenEvent		"%sessionId: %user@%remoteAddress %action %object %outcome (listen on: %interface:%port)"	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address %action: ‘forward’ %object: ‘tcpip-forward’ %outcome: ‘granted’ or ‘denied’ or ‘failed’ %interface: local bind address %port: local port
20	TimeoutEvent		"%sessionId: %user@%remoteAddress %action %object"	%sessionId: SESSION-LOG-ID %remoteAddress: remote IP address %action: ‘idle timeout’ %object: module experiencing timeout (currently always ‘SFTPSERV’)
21	SftpServerFatalErrorEvent		“%sessionId: %user@%remoteAddress %action %object error info: '%errInfo', %processType process %processName stopping..."	%sessionId: SESSION-LOG-ID %user: SSH username %remoteAddress: remote IP address

Event Id	Event Name	Conditions	Pattern	Token Values
				%action: ‘terminate’ %object: ‘SFTP process’ %errInfo: error detail %processType: ‘SFTPSERV’ %processName: process name

Log File/Audit File Rollover

When logging to a file, SSH2 uses a round-robin mechanism to switch to a new file. Log file rollover applies both to auditing (to the file configured with the AUDITFILE parameter) and logging (to the file configured with the LOGFILE parameter).

A log file rollover occurs when the logfile is greater than the size configured in the parameter LOGMAXFILELENGTH or when the audit file is greater than the size configured in the parameter AUDITMAXFILELENGTH. It is also possible to force the rollover via SSHCOM command (see ROLLOVER AUDITFILE and ROLLOVER LOGFILE in chapter "SSHCOM Command Reference".

SSH2 implements a log file round-robin with at least 10 files. The number of files can be configured using the LOGFILERETENTION (or AUDITFILERETENTION) parameter. If the number of retention files is set to 0 (LOGFILERETENTION or AUDITFILERETENTION), then the content of file configured via LOGFILE (or AUDITFILE) will be purged as soon as the file size reaches the maximum configured size. But it is recommended to use at least 10 retention files.

Archive files generated during rollover will be created by appending a number to the log file name. The number of digits of the number appended will be calculated depending on the number of files to keep.

With LOGFILERETENTION set to 10 (the default value), the archive files for a LOGFILE of SLOG will be called SLOG0, SLOG1, ... SLOG9.

With LOGFILERETENTION set to 1000, the archive files for a LOGFILE of SLOG will be called SLOG000, SLOG001, ... SLOG999.

Viewing File Contents from Guardian with SHOWLOG

SSH2 servers may be configured to write log or audit files to disk. For performance reasons, those log files are created as unstructured files:

```
15> fileinfo SSH2log
$data1.comfSSH2
              CODE              EOF   LAST MODIFIED   OWNER RWEP   PExt   Sext
SSH2log         0              5044 25sep2003 15:14 110,111 aaaa      4     28
16>
```

While the program is running, the log file is always open, however it may be concurrently opened for viewing. To convert the unstructured file into a readable format, a tool SHOWLOG is supplied. Invoking SHOWLOG without arguments will display a brief syntax summary:

```
20> run showlog
SHOWLOG log file converter Version T9999A06_15Nov2012_HP_SHOWLOG_0024
usage: SHOWLOG <log-file> [<out-file> [<start> [<end>]]]
    <log-file> | the input log file to be converted
    <out-file> | file to write to, default is '*' meaning the home terminal
    <start>    | either byte offset from beginning or a timestamp
    <end>      | either number of bytes after beginning or a timestamp

-- Supported timestamp formats: --
    "ddmmmyy HH:MM:SS.TTT", "ddmmmyy HH:MM:SS.TT", "ddmmmyy HH:MM:SS",
    "ddmmmyy HH:MM", "ddmmmyy", "HH:MM:SS.TTT", "HH:MM:SS.TT", "HH:MM:SS",
    "HH:MM".
    Current date is used if date not specified as part of <start> timestamp.
    Date from <start> is used if date not specified in <end> timestamp.

-- Examples --
Whole log file written to home terminal:
     SHOWLOG logfile
Display 1000 bytes starting at offset 10000 written to EDIT file logedit
     SHOWLOG logfile logedit 10000 1000
Starting at offset 200000 and display all bytes up to the end of the file
     SHOWLOG logfile * 200000
Display messages in timeframe to home terminal
     SHOWLOG logfile * "03Jan11 03:15" "05Jan07 21:30:10.89"
Write messages in timeframe to EDIT file logedit starting from specified time
     SHOWLOG logfile logedit "01Feb12 01:02:03.67"
21>
```

If SHOWLOG is run with only the name of the log file as first runtime argument, it will dump the whole log file to the home terminal. The byte offset within the log file will be displayed every now and then; this allows you to limit the output of SHOWLOG to certain sections of the log file as shown below.

```
$US SSH92 33> run showlog sh54log
SHOWLOG log file converter Version T9999A05_16Apr2009_HP_SHOWLOG_0022
---processing in-file 'sh54log'
$SSH54|18Apr12 17:07:30.97|20|-------------------------------------------------------
--------
$SSH54|18Apr12 17:07:31.00|10|SSH2 version T9999H06_17Apr2012_comForte_SSH2_0092
$SSH54|18Apr12 17:07:31.02|10|config file:  '(none)'
$SSH54|18Apr12 17:07:31.03|20|object filename is  '\NPNS01.$US.SSH92.SSH2'
$SSH54|18Apr12 17:07:31.04|20|object subvolume is '\NPNS01.$US.SSH92', priority is 11
$SSH54|18Apr12 17:07:31.06|20|dumping configuration:
[def  ] ALLOWEDAUTHENTICATIONS <keyboard-interactive,password,publickey>
[par  ] ALLOWEDSUBSYSTEMS    <sftp,tacl>
[par  ] ALLOWFROZENSYSTEMUSER <TRUE>
[def  ] ALLOWINFOSSH2        <ALL>
[def  ] ALLOWPASSWORDSTORE   <TRUE>
[run  ] ALLOWTCPFORWARDING   <true>
[par  ] AUDITCONSOLE         <%;>
[run  ] AUDITFILE            <SH54AUD>
[def  ] AUDITFILERETENTION   <10>
```

```
[def  ] AUDITFORMAT          <21>
[def  ] AUDITMAXFILELENGTH   <20000>
[def  ] AUTOADDAUTHPRINCIPAL <FALSE>
[run  ] AUTOADDSYSTEMUSERS   <TRUE>
[run  ] AUTOADDSYSTEMUSERSLIKE <templateuser>
[def  ] BACKUPCPU            <NONE>
[def  ] BANNER               <*>
[def  ] BURSTSUPPRESSION     <FALSE>
[def  ] BURSTSUPPRESSIONEXPIRATIONTIME <300>
[def  ] BURSTSUPPRESSIONMAXLOGLEVEL <40>
[def  ] CACHEBURSTSUPPRESSION <TRUE>
[def  ] CIPCOMPATERROR       <*>
[def  ] CIPHERS              <aes256-cbc,twofish256-cbc,twofish-cbc,aes128-
cbc,twofish128-cbc,blowfish-cbc,3des-cbc,arcfour,cast128-
cbc>
[def  ] CLIENTALLOWEDAUTHENTICATIONS <none,gssapi-with-
mic,publickey,password,keyboard-interactive>
[par  ] CLIENTMODEOWNERPOLICY <GUARDIAN>
[def  ] COMPRESSION          <TRUE>
[def  ] CONFIG               <>
[def  ] CONFIG2              <*>
[def  ] CONSOLEBURSTSUPPRESSION <FALSE>
[def  ] CPUSET               <>
[par  ] CUSTOMER             <comForte GmbH>
[run  ] DISCONNECTIFUSERUNKNOWN <FALSE>
[def  ] EMSBURSTSUPPRESSION  <FALSE>
[def  ] ENABLESTATISTICSATSTARTUP <FALSE>
[def  ] FILEBURSTSUPPRESSION <FALSE>
[def  ] FULLSSHCOMACCESSGROUP1 <>
[def  ] FULLSSHCOMACCESSUSER1 <>
[run  ] GSSAUTH              <$GSSy>
[def  ] GSSGEXKEX            <FALSE>
[def  ] GSSKEX               <TRUE>
[def  ] GUARDIANATTRIBUTESEPARATOR <,>
[def  ] HOSTKEY              <HOSTKEY>
[expl ] INTERFACE            <0::0>
[def  ] INTERFACEOUT         <0::0>
[def  ] INTERVALLIVEPRIVATEUSERKEY <730>
[def  ] INTERVALLIVEPUBLICUSERKEY <730>
[def  ] INTERVALPENDINGPRIVATEUSERKEY <0>
[def  ] INTERVALPENDINGPUBLICUSERKEY <0>
[par  ] IPMODE               <DUAL>
[def  ] LICENSE              <\NPNS01.$US.SSH92.LICENSE>
[par  ] LIFECYCLEPOLICYPRIVATEUSERKEY <FIXED>
[par  ] LIFECYCLEPOLICYPUBLICUSERKEY <FIXED>
[def  ] LOGCACHEDUMPONABORT  <TRUE>
[par  ] LOGCACHESIZE         <500000>
[def  ] LOGCONSOLE           <%>
[run  ] LOGEMS               <$USLOG>
[def  ] LOGEMSKEEPCOLLECTOROPENED <TRUE>
[run  ] LOGFILE              <SH54LOG>
[def  ] LOGFILERETENTION     <10>
[def  ] LOGFORMATCONSOLE     <93>
[def  ] LOGFORMATEMS         <16>
[def  ] LOGFORMATFILE        <93>
[run  ] LOGFTPSCONSOLE       <%>
[def  ] LOGLEVELCACHE        <50>
[run  ] LOGLEVELCONSOLE      <88>
[run  ] LOGLEVELEMS          <70>
[run  ] LOGLEVELFILE         <50>
[def  ] LOGMAXFILELENGTH     <20000>
[def  ] MACS                 <hmac-sha1,hmac-md5,hmac-sha1-96,hmac-md5-96>
[def  ] PARTIALSSHCOMACCESSGROUP1 <>
[def  ] PARTIALSSHCOMACCESSUSER1 <>
[run  ] PORT                 <54022>
[par  ] PTCPIPFILTERKEY      <SSH48>
[run  ] PTYSERVER            <$PTY54>
[def  ] RECORDDELIMITER      <LF>
[def  ] RESTRICTIONCHECKFAILEDDEFAULT <FALSE>
[par  ] SFTPALLOWGUARDIANCD  <TRUE>
[def  ] SFTPCPUSET           <>
```

```
[par  ] SFTPEDITLINEMODE     <cut>
[def  ] SFTPEDITLINENUMBERDECIMALINCR <1000>
[def  ] SFTPEDITLINESTARTDECIMALINCR <-1>
[par  ] SFTPEXCLUSIONMODEREAD <EXCLUSIVE>
[def  ] SFTPIDLETIMEOUT      <-1>
[def  ] SFTPMAXEXTENTS       <900>
[def  ] SFTPPRIMARYEXTENTSIZE <2>
[def  ] SFTPREALPATHFILEATTRIBUTEECHOED <FALSE>
[def  ] SFTPSECONDARYEXTENTSIZE <100>
[def  ] SFTPUPSHIFTGUARDIANFILENAMES <FALSE>
[def  ] SHELLENVIRONMENT     <>
[def  ] SOCKETKEEPALIVE      <1>
[par  ] SOCKETRCVBUF         <122880>
[par  ] SOCKETSNDBUF         <122880>
[def  ] SOCKTCPMAXRXMT       <0>
[def  ] SOCKTCPMINRXMT       <0>
[def  ] SOCKTCPRXMTCNT       <0>
[def  ] SOCKTCPTOTRXMTVAL    <0>
[def  ] SSHAUTOKEXBYTES      <1073741824>
[run  ] SSHAUTOKEXTIME       <60>
[def  ] SSHCTL               <SSHCTL>
[def  ] SSHCTLAUDIT          <TRUE>
[def  ] SSHKEEPALIVETIME     <60>
[def  ] STOREDPASSWORDSONLY  <FALSE>
[run  ] STRICTHOSTKEYCHECKING <false>
[run  ] SUBNET               <$ZSAM1>
[def  ] SUPPRESSCOMMENTINSSHVERSION <FALSE>
[def  ] TCPIPHOSTFILE        <*>
[def  ] TCPIPNODEFILE        <*>
[def  ] TCPIPRESOLVERNAME    <*>
$SSH54|18Apr12 17:07:31.17|10|CRYPTOPP version
T9999H06_12Apr2012_comForte_CRYPTOPP_0023
$SSH54|18Apr12 17:07:31.21|10|SSH config database SSHCTL opened.
$SSH54|18Apr12 17:07:31.23|20|parameter SUBNET was evaluated
$SSH54|18Apr12 17:07:31.24|20|DEFINE =TCPIP^PROCESS^NAME was set to <\NPNS01.$ZSAM1>
$SSH54|18Apr12 17:07:31.25|20|TCP/IP process is $ZSAM1
$SSH54|18Apr12 17:07:31.25|20|DEFINE =PTCPIP^FILTER^KEY was set to
<\NPNS01.$US.SSH92.SSH48>
$SSH54|18Apr12 17:07:3
```

The second runtime argument can be used to create a new EDIT file containing the log file contents. The following example shows how to convert the whole log file into an edit file (note that this can take some time for large files):

```
42> run showlog SSH2log logedit
SHOWLOG log file converter Version T9999A05_16Apr2009_HP_SHOWLOG_0022
writing out-file 'logedit'
---processing in-file 'ssh2log'




---
--- EOF reached, done
---
43> fileinfo logedit

$US.SSH89C

              CODE              EOF   LAST MODIFIED  OWNER  RWEP   PExt   SExt
LOGEDIT        101             6086 23DEC2010 17:36 255,255 NONO     14     2844>
```

The third and last runtime argument can be used to limit the part of the file that is converted. This is helpful for the viewing large log files. The following example illustrates the dumping of a large log file. Only a limited number of log messages (totaling 10.000 bytes) after a given offset (5.000.000) are shown:

```
33> run showlog SSH2log * 5000000 10000
SHOWLOG log file converter Version T9999A05_16Apr2009_HP_SHOWLOG_0022
starting at offset 5000000
dumping at most 10000 bytes
---processing in-file 'SSH2log'
```

```
(output not shown here)

---
---finishing dump of file before end-of-file
---

---done 34>
```

Notes

- In this example, by using '*' as the second runtime argument, the output is written to the home terminal. When using the byte offset parameter or the byte offset and length parameter, the out file parameter must be specified as well.
- Starting with SPR T0801^ABE, SHOWLOG reports errors regarding invalid timestamps. It is now possible to just specify a time without a date. If there is only a time for the <start> timestamp, then the current day is used as default. If there is no date part for the <end> timestamp, then the day of the <start> timestamp is used as default for the <end> date.
  It is now also possible to use a comma as delimiter between date and time part, which allows dropping the double quotes that are necessary if space is used as delimiter.
  SHOWLOG now accepts one digit hours and days as in "1Nov12,3:10" which is treated as "01Nov12,03:10".

Viewing File Contents from OSS

The log or audit files created by SSH2 are unstructured files and can be viewed from OSS with standard OSS tools such as more or tail. Standard OSS filter tools such as grep, awk, or wc can also be applied. This allows users to make use of the powerful Unix syntax for doing text processing.

Performance Considerations

Introduction

As the saying goes, "there is no such thing as a free lunch": using SSH2 to encrypt traffic will consume some CPU cycles on your NonStop host. The natural question "how much CPU resources does encryption consume" has no simple answer, it will depend on many factors:

- In general:
    - How many SSH connections are created—the initial setup of an SSH session involves a public-key operation, which require some CPU intensive calculations.
    - The key sizes used for the public/private key pairs both on the host and on the client—using a more secure 1024 bit key pair will cause more overhead for the initial setup than a 512 bit RSA key pair.
    - The selected cipher for bulk encryption—for example, a cipher using 168 bit 3DES will consume more CPU cycles than a 128 bit ARCFOUR based cipher suite.
- For SFTP traffic:
    - The throughput of the transmitted data. How many files of which size are transmitted in which time?
    - Type of data read (structured or non-structured files)
    - The SFTP client used and the system it is run on.
    - Speed of file listings depends on the way an SFTP client makes use of the file attributes received from the SFTP server.

So there is no general answer to the question; the answer will depend on your individual system use.

However, measurements show that today's NonStop systems aren’t as bad in number crunching (and that’s what encrypting and decrypting is basically about) as one would think.

The following sections will show the results of some selected measurements. The conclusions drawn from these can be used to estimate what performance behavior you can expect on your system.

Note: All measurements referred to in this chapter have been performed on a 2 processor S7600. HP provides performance metrics that allow you to extrapolate those results to other systems. These metrics can be provided upon request.

Performance Analysis of SSH Session Establishment

Performance Running as SSH Daemon

The performance impact of the initial SSH session setup should be viewed separately. As explained before, establishing an SSH session involves several CPU-intensive public key operations. The amount of CPU cycles consumed depends upon the key sizes used.

The following table shows the CPU consumption of an SSH session setup (without any data transfer taking place) for a DSA host key with 1024 bit length and for RSA client keys with the sizes as stated in the table:

Client Key size [bits]	Approximate CPU consumption [milliseconds]
512	234
1024	236
2048	242

It is very hard to predict future developments, both in cryptography and computer technology, which makes it next to impossible to tell in advance what key size will be sufficient in the years to come. We recommend using a key size of 1024 bits for the time being.

Performance Analysis of SFTP Traffic

To get an indication of the performance of the SSH2 component and the subordinate SFTPSERV processes when acting as SFTP daemon, the average transfer rate and CPU consumption has been measured while a file with 50 MB of data has been transferred via SFTP.

The following table shows the result of the measurement:

Partner system	Direction of transfer	Cipher Suite/MAC algorithm	Time elapsed [s]	CPU time used [s]	Through-put [KB/s]	CPU ms/MB transfer	CPU usage
Linux, OpenSSH	NonStop to Partner system	AES-128/MD5	66,5	27,1	734	568	41 %
Linux, OpenSSH	Partner system to NonStop	AES-128/MD5	242	26,6	202	557	11%

Please bear in mind that the measured transfer rate does not only depend on the performance of the SSH2/SFTPSERV components, but also on the network throughput and the performance of the remote SFTP client or server.

The most significant column of the table probably is the value "CPU ms/MB transfer" which should give a good estimate for the CPU milliseconds needed to transfer one Megabyte of data using SFTP.

SFTPSERV Performance of ls Command with Wildcards

The output from command ls (list) can be delayed when wildcards are used and the file information returned by SFTPSERV is not processed effectively. Unlike the ftp protocol the sftp protocol does not define two commands for listing the names of files in a directory (ftp: NLST) and listing of all file attributes of files in a directory (ftp: LIST). There is only one command in the sftp protocol (READDIR) that always retrieves all attributes of the files in a directory. In case of a wildcard (e.g. ls test*) the SFTP client will do the pattern matching after all file attributes have been retrieved from the SFTP server. After the pattern matching the SFTP client could display the file listing but there are

SFTP clients that retrieve the file attributes for each file matching the specified pattern again from the SFTP server. This is causing unnecessary overhead. If the delay is of unacceptable length, the following workarounds may help:

- Reduce the number of files in one directory/subvolume on NonStop
- Set USER attribute SFTP-GUARDIAN-FILESET if information of files in a Guardian subvolume is listed. In this way the pattern matching is done on the server and the data being sent to the client can be greatly reduced. Different patterns can be defined by using different ssh user records with the same SYSTEM-USER.

Performance When Running as SSH Client

The above measurements have been repeated with the SFTP client now running on the NonStop system. The following table shows the result of the measurement:

Partner system	Direction of transfer	Cipher Suite/MAC algorithm	Time elapsed [s]	CPU time used [s]	Through-put [KB/s]	CPU ms/MB transfer	CPU usage
Linux, OpenSSH	NonStop to Partner system	AES-128/MD5	54	26,2	904	549	48 %
Linux, OpenSSH	Partner system to NonStop	AES-128/MD5	238	28,0	205	586	12 %

Summary

There is no answer to the seemingly simple question: "How much CPU cycles will 128 bit encryption consume on my system?" To understand why, consider asking an automobile expert the question, "How much fuel will I need for my vacation?" (Without giving away more information.) Regardless of how much the expert knows about cars and engines, he will not be able to give an answer unless you tell him such information as…

- The maker of the car.
- Where you want to go.
- Your driving habits.

Using the data provided in this chapter should allow you to get an estimate of the CPU resources that should be utilized by SSH2 within your specific environment.

Troubleshooting

Introduction

This chapter lists the information items needed by support when reporting an SSH2 related problem and a number of common error messages that SSH2 or an SSH client can produce, and explains what they mean in more detail.

We do not attempt to list all error messages here: there are many that should never occur, and some that should be self-explanatory.

Information Needed By Support

When sending a support request, please provide the following information (the more information you supply, the better support can be provided):

- Short description (one or two lines)
- Product Environment
    - SSH2 Version: Please run the SSH2INFO macro on your SSH2 installation subvolume and send the result.
    - SSH2 Status: If possible, please run SSHCOM against a running instance of the SSH2 process, execute the INFO SSH2 command and send the output.
    - Clients/Servers: Which SSH/SFTP clients and daemons are communicating with the NonStop™ platform via SecurFTP/SSH? Please provide platform information, product names and version numbers.
- Problem Description
    - Detailed description: Please describe the problem (expected versus observed behavior).
    - Context: "Installing the product and having a problem getting it to work" or "Product has been running successfully; this is a new issue" or any other detail describing the context.
    - Frequency: How often does the problem occur? (sporadically/frequently/always)
    - Occurrence: Where does the problem occur? (on all workstations or sessions/only on selected workstations or session)
    - Error Message: Is there an error message generated? Please specify the exact text. The error message may be taken from EMS, from a log file or captured from a screen.
    - Reproduction: Please describe the exact steps that led to the problem.

General SSH2 Error Messages

Errors that impact the operation of the SSH2 process are reported as error logs or warning messages. Log messages are written to SSH's log destinations as configured by the LOGCONSOLE, LOGFILE and LOGEMS parameters.

Error log messages have a log level of 10.

```
unexpected exception: <error detail>. SSH2 terminating.
```

<error detail>

Describes the error condition.

Cause: The SSH2 process encountered a fatal error condition.

Effect: The SSH2 process terminates.

Recovery: Any corrective action depends on <error detail>.

```
Invalid runmode. SSH2 terminating.
Valid runmodes are CLIENT, DAEMON, SERVER (same as DAEMON), ADMIN, NOADMIN or ALL.
```

Cause: The SSH2 process was started with an invalid run mode.

Effect: The SSH2 process terminates.

Recovery: Use a valid run mode.

```
Failed to <operation> private host key file <key file name>
```

<operation>

Is either "create" or "write".

<key file name>

Is the name of the private host key file as given by the HOSTKEY parameter.

Cause: SSH2 could not create or write the private host key file.

Effect: The SSH2 process continues processing with the generated private key. As the key could not be stored, the host key will change after restart of SSH2 (SSH2 will generate a new key).

Recovery: Check the HOSTKEY parameter if it refers to a valid file name. You may also need to check your SAFEGUARD settings to ensure SSH2 is authorized to create or write the HOSTKEY file.

```
Error loading private host key: <error detail>
```

Cause: SSH2 could not load the private host key from the HOSTKEY file.

Effect: The SSH2 process terminates.

Recovery: Validate that the file referred to by the HOSTKEY parameter contains a private key previously generated by SSH2.

```
Info ProtectionRecord: Processing OBJECTTYPE USER access configuration:
   ignoring entry <entry> because type <type-num> (<type-name>) is REMOTE specific
```

Cause: SSH2 found an OBJECTTYPE USER entry with network id

Effect: SSH ignores that entry

Recovery: Add a local ACL OBJECTTYPE USER entry, i.e. one without \node-spec.

Session Related SSH2 Errors

Session related errors are reported as SSH2 warning log messages. Warning messages have a log level of 20.

Session Related Error Messages of SSH2 Daemon

All messages related to a connection received by a remote SSH client are preceded by a session ID. These messages adhere to the following format:

```
<session id> := <remote IP address>:<remote port>
```

<remote ip address> is the IP address of the system the SSH client is connecting from and <remote port> is the port number assigned to the SSH client session on the remote side.

The messages are as follows:

```
<session id>: Error: <error description>
```

<error description>

Is a description of the error condition.

Cause: An error occurred on the SSH session. Typical errors include network related errors.

Effect: The SSH session is closed.

Recovery: Any corrective action depends on <error description>.

```
<session id>: Disconnect from remote: <disconnect reason>
```

<disconnect reason>

Is a description received from the remote client to describe the reason for disconnecting.

Cause: The SSH client gracefully terminated the SSH session.

Effect: The SSH session is closed.

Recovery: Any corrective action depends on <disconnect reason>. It may be required on the remote SSH client side. Contact the comForte support, if <disconnect reason> indicates an SSH protocol error.

```
<session id>: User auth method mismatch, available: <remaining methods>, <requested
method>
```

<remaining methods>

List of SSH authentication methods that are supported by SSH2 that have not been tried by the SSH client.

<requested method>

Authentication method requested by the SSH client.

Cause: The SSH client tried to use an authentication method not supported by SSH2.

Effect: The remote SSH user cannot be authenticated.

Recovery: Configure an authentication method for SSH client that is supported by SSH2, e.g. "public key" authentication.

```
<session id>: Authentication of user <user name> failed: <error detail>
```

<user name>

Name of the remote user.

<error detail>

Describes the reason for the authentication failure.

Cause: An error occurred during the authentication of the user. Typical errors are:

"User not found": <user name> does not exist in the SSHCTL.

"User is frozen": <user name> exists in the SSHCTL but is frozen.

Effect: The remote SSH user cannot be authenticated. The session will be terminated.

Recovery: Any corrective action depends on the reason for the authentication failure. It may be required to add, correct or thaw a user name using SSHCOM.

```
<session id>: No more authentication requests possible for <user name>
```

<user name>

Name of the remote user.

Cause: The maximum number of authentication requests exceeded. Typically, this condition can occur with password authentication, if the SSH clients sends an invalid password for three times.

Effect: The remote SSH user cannot be authenticated. The session will be terminated.

Recovery: Use correct credentials for the user with the SSH client.

```
<session id>: password change for user <user name> failed: <error detail>
```

<user name>

Name of the remote user

<error detail>

is a description of the error that made the password change fail.

Cause: An error occurred when trying to change the user’s password, upon request of the SSH client.

Effect: The password could not be changed.

Recovery: Any corrective action depends on cause.

```
<session id>: public key authentication failed, algorithm not supported.
```

Cause: The SSH client tried to use an algorithm for public key authentication that is not supported by SSH2.

Effect: The password could not be changed

Recovery: Configure the SSH client to use a public key algorithm supported by SSH2.

```
<session id>: public key authentication failed, too many keys
```

Cause: The SSH client has more than ten public keys that did not match any public key stored for the user in the SSHCTL.

Effect: The public key authentication is aborted. The user cannot be authenticated.

Recovery: Reduce the number of identities (private keys) for the user presented by the SSH client. Usually, this involves adding fewer keys to an SSH agent.

```
<session id>: public key authentication failed, invalid signature
```

Cause: The signature presented by the SSH client does not match the public key.

Effect: The authentication is rejected.

Recovery: Check the SSH client that presented the invalid signature.

```
<session id>: <authentication method> for user <user name> not supported
```

<authentication method>

Is the authentication method requested by the SSH client

<user name>

Is the name of the remote user.

Cause: The SSH client requested an authentication method that is not supported by SSH2 or has been disallowed for this user.

Effect: The authentication is rejected.

Recovery: Use a supported authentication method with the SSH client. Check the settings for this user in the SSH2 user base.

```
<session id>: channel request for subsystem sftp denied
```

Cause: SFTP is administratively disallowed for this user.

Effect: The channel request for the SFTP subsystem is rejected.

Recovery: Have the SSH client not use SFTP or grant SFTP access by setting the SFTP-SECURITY attribute for the user to a value other than NONE.

```
<session id>: SFTPSERV process initialisation failed, could not chdir or chroot to
user's SFTP-INITIAL-DIRECTORY, error <error number>
```

<error number>

Is the error number that was raised by the chdir or chroot operation.

Cause: Chdir or chroot failed when setting the user’s SFTP-INITIAL-DIRECTORY. A possible reason is that the directory does not exist.

Effect: The channel request for the SFTP subsystem is rejected.

Recovery: Check the setting of SFTP-INITIAL-DIRECTORY for the relevant user.

```
<session id>: could not launch program <program name>, error <error number>, detail
<detail error number>
```

<program name>

Is the name of the program file that SSH2 tried to start.

<error number>

Is the error number that was raised by the PROCESSCREATE function.

<error number detail>

Is the detail error number that was raised by the PROCESSCREATE function.

Cause: PROCESSCREATE failed with an error

Effect: The channel request (e.g. subsystem SFTP) fails which the process (e.g. SFTPSERV) should be created for.

Recovery: Check the NonStop™ server documentation for PROCESSCREATE error descriptions. If SFTPSERV could not be started make sure the program is located in the same directory as SSH2.

```
<session id>: SFTPSERV process initialisation failed, error <error number> during
startup procedure
```

<error number>

Is the error number that was raised during the initialization of the SFTPSERV process.

Cause: An error occurred during the initial inter process communication with the SFTPSERV process.

Effect: The channel request for the SFTP subsystem is rejected.

Recovery: Check if SFTPSERV abended during the initialization procedure. Contact comForte if this problem persists.

```
<session id>: forwarding from <host>:<port> to <target-host>:<target-port> denied
```

<host>

Is the IP address of the socket client the SSH client tries to forward a connection from.

<port>

Is the IP address of the socket client the SSH client tries to forward a connection from.

<target-host>

Is the IP address the SSH client requested to forward the connection to.

<target-port>

Is the port number the SSH client requested to forward the connection to.

Cause: An SSH client requested the forwarding of a connection. However, this has been administratively prohibited, e.g. by setting the ALLOWTCPFORWARDING parameter to FALSE.

Effect: The forwarding request is rejected.

Recovery: If forwarding is desired, check the setting of ALLOWTCPFORWARDING.

```
<session id>: forwarding <protocol> connection from <host>:<port> to <target-
host>:<target-port>  failed (<error detail>)
```

<host>

Is the IP address of the socket client the SSH client tries to forward a connection from.

<port>

Is the IP address of the socket client the SSH client tries to forward a connection from.

<target-host>

Is the IP address the SSH client requested to forward the connection to.

<target-port>

Is the port number the SSH client requested to forward the connection to.

<error detail>

Describes the error that occurred.

Cause: An error occurred when trying to forward a connection.

Effect: The forwarding request fails.

Recovery: Any corrective action depends on <error detail>. A typical error is a failure to connect to the target host and port. The SSH client may need to correct its port forwarding configuration.

```
<session id>: listen request on <interface>:<port> denied
```

<interface>

Is the IP address of the local interface the SSH client tries to establish a listen for.

<port>

Is the port number SSH client tries to listen on.

Cause: The SSH client tried to establish a remote port forwarding with the SSH2 server. However, this has been administratively prohibited, e.g. by setting the ALLOWTCPFORWARDING parameter to FALSE.

Effect: The forwarding request is rejected.

Recovery: If forwarding is desired, check the setting of ALLOWTCPFORWARDING.

```
<session id>: remote forwarding request failed, server could not listen on
<interface>:<port> (<error detail>)
```

<interface>

Is the IP address of the local interface SSH client tries to establish a listen for.

<port>

Is the port number SSH client tries to listen on.

<error detail>

Describes the error that occurred.

Cause: An error occurred when trying to establish a listen for remote port forwarding

Effect: The remote port forwarding request fails.

Recovery: Any corrective action depends on <error detail>. A typical error is a failure to bind to the given port. The SSH client may need to correct its port forwarding configuration.

Session Related Messages of SSH2 in Client Mode

All SSH2 messages related to an outgoing connection to a remote SSH daemon initiated by a NonStop client process (e.g. SFTP, SFTPOSS) are preceded by a session ID. These messages adhere to the following format:

```
<session id> := <process id>
```

<process name> is the name of the NonStop client process initiating the SSH connection.

```
<session id>: client access to known host <known host name> denied, host is frozen
```

<known host name>

Is the name of a KNOWNHOST entity contained in the SSHCTL.

Cause: The SSH client (e.g. SFTP) tried to access a known host that was frozen.

Effect: The client access to the host is denied. The client connection fails.

Recovery: If access to the host is desired, use the SSHCOM THAW KNOWNHOST command to thaw the host.

```
<session id>: client access to known host <known host name> denied, public key changed
```

<known host name>

Is the name of a KNOWNHOST entity contained in the SSHCTL.

Cause: The public key of the host the SSH client (e.g. SFTP) tried to access does not match the public key stored for the KNOWNHOST in SSHCTL. Important note: THIS COULD BE CAUSED BY a "man-in-the-middle" attack.

Effect: The client access to the host is denied. The client connection fails.

Recovery: Check if the identity of the target host has really been changed. If access to the host is desired, use the SSHCOM ALTER KNOWNHOST command to alter the public key of the host.

```
<session id>: client access to unknown host at <host>:<port> denied
```

Cause: The public key of the host the SSH client (e.g. SFTP) tried to access does not match the public key stored for the KNOWNHOST in SSHCTL. Important note: THIS COULD BE CAUSED BY a "man-in-the-middle" attack.

Effect: The client access to the host is denied. The client connection fails.

Recovery: Check if the identity of the target host has really been changed. If access to the host is desired, use the SSHCOM ALTER KNOWNHOST command to alter the public key of the host.

```
<session id>: exception during host verification: <error detail>
```

<error detail>

Is a description of the error condition.

Cause: An unexpected error occurred during the verification of the host the SSH client (e.g. SFTP) connected to. For example, this could be caused by a problem with accessing the SSHCTL database.

Effect: The client access to the host is denied. The client connection fails.

Recovery: Any corrective action depends on error detail.

```
<session id>: Authentication failed
```

Cause: The authentication of the user with the remote SSH server failed.

Effect: The client access to the host is denied. The client connection fails.

Recovery: Additional error information is returned to the SSH client (e.g. SFTP). Check the user's credentials (private keys or password) for accuracy. Check if any of the user's private keys are made known to the SSH server.

```
<session id>: failed to open channel, reason: <reason>
```

<reason>

Is a description of the cause of failure, which is sent by the remote SSH server.

Cause: The remote SSH server could not open the channel the local SSH client requested to open.

Effect: The channel is not opened.

Recovery: Any corrective action depends on <reason>.

```
<session id>: channel request failed
```

Cause: The remote SSH server reports a failure of a channel request previously issued for the local SSH client. For example the "subsystem sftp" channel request may have failed.

Effect: The channel is not opened.

Recovery: Check the remote SSH server installation.

```
<session id>: error on channel: <error description >
```

<error description>

Describes the error.

Cause: An error occurred on the SSH channel.

Effect: The SSH channel is closed.

Recovery: Any corrective action depends on <error description>.

```
<session id>: error on ssh session: <error description>
```

<error description>

Describes the error.

Cause: An error occurred on the SSH session. Typical errors are network related.

Effect: The SSH session is closed.

Recovery: Any corrective action depends on <error description>.

Client Error Messages

This section describes common errors generated by the SSH[OSS] and SFTP[OSS] client programs.

```
could not open SSH2 process : <error detail>
```

<error detail>

Describes the error condition.

Cause: The client failed to open a suitable SSH2 server process.

Effect: The client process terminates.

Recovery: Check if any SSH2 processes are started.

```
connect failed, error <error detail>
```

<error detail>

Describes the error condition.

Cause: The client could not establish the TCP connection to the remote host. Typical causes are:

Message	Meaning
Socket: Connect operation failed with error 4127	the remote host refused the connection
Socket: gethostbyname operation failed with error 4022	The host name could not be resolved

Effect: The client process terminates.

Recovery: Any corrective action depends on <error detail>.

```
WARNING: REMOTE HOST IDENTIFICATION UNKNOWN!
The host public key fingerprint is
  babble: <bubble-babble>
  MD5:    <md5>
```

<bubble-babble>

Is the "bubble-babble" fingerprint of the remote host’s public key.

<MD5>

Is the "bubble-babble" fingerprint of the remote host’s public key.

Cause: The client failed to open a suitable SSH2 server process.

Effect: Depends on the configuration of the STRICTHOSTKEYCHECKING parameter of the SSH2 process serving this client.

If STRICTHOSTKEYCHECKING is FALSE, the client will display the following prompt:

```
Continue and add the host to the knownhost store(yes/no)?
```

If the user enters "yes", a KNOWNHOST object storing the remote host’s public key is automatically added for the user to the SSHCTL database. Otherwise, the client process terminates.

If STRICTHOSTKEYCHECKING is FALSE, the client will display the following messages:

```
For convenience the host identification has been added FROZEN.
Host name is <hostname>
```

```
Please contact your system administrator.
```

In this case, SSH2 has automatically added a KNOWNHOST object named <hostname>, storing the remote host's public key. However, the KNOWNHOST attribute FROZEN is set to disallow any connections to that host until it is THAWED.

Recovery: To allow access to the host, which has been added FROZEN to the SSHCTL, you can use the following SSHCOM command:

```
THAW KNOWNHOST <hostname>
```

```
ERROR: REMOTE HOST IDENTIFICATION IS FROZEN!
Frozen host is <hostname>
```

<hostname>

Is the name of the KNOWNHOST object holding the remote host's public key.

Cause: The KNOWNHOST object holding the remote host's public key is FROZEN.

Effect: The client process terminates.

Recovery: To allow access to the host, which has been set FROZEN, you can use the following SSHCOM command:

```
THAW KNOWNHOST <hostname>
```

```
ERROR: REMOTE HOST IDENTIFICATION HAS CHANGED!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the host key has just been changed.
The fingerprints for the key sent by the remote host are:
  babble: <bubble-babble>
  MD5:    <md5
Offending key is <keyname>
Please contact your system administrator.
```

<bubble-babble>

Is the "bubble-babble" fingerprint of the remote host's public key.

<MD5>

Is the "bubble-babble" fingerprint of the remote host's public key.

<keyname>

Is the name of the KNOWNHOST object holding the remote host's public key.

Cause: The remote host's public key does not match the key stored in the KNOWNHOST object for this IP address and port number. This can happen if the remote SSH daemon has changed its public key. It can also be caused by a man-in-the-middle attack.

Effect: The client process terminates.

Recovery: You should ensure that the error is caused by a legitimate change of the remote host's key. If the error is not caused by eavesdropping, you should update the KNOWNHOST referring to the remote host. This can be done as follows:

a) Obtain the remote host's new public key or public key fingerprint and update the relevant KNOWNHOST using SSHCOM as follows:

```
ALTER KNOWNHOST <keyname>, PUBLICKEY ...
```

b) Using SSHCOM, delete the existing KNOWNHOST entry as follows:

```
DELETE KNOWNHOST <keyname>
```

After reconnecting the client, a "WARNING: REMOTE HOST IDENTIFICATION UNKNOWN!" will be issued and a new KNOWNHOST entry for the remote host’s new public key is automatically added to the SSHCTL. If the SSH2 parameter STRICTHOSTKEYCHECKING is TRUE, then you need to thaw the newly added KNOWNHOST entry to establish a connection:

```
THAW KNOWNHOST <hostname>
```

```
Couldn’t read packet: <error detail>
Couldn’t write packet: <error detail>
```

<error detail>

Describes the error condition.

Cause: The client failed to receive/send a packet from/to the SSH2/SFTP channel. Typical causes are that the remote SSH server has terminated the SSH session of SFTP channel.

Effect: The client process terminates. Any ongoing file transfer will be aborted.

Recovery: Any corrective action depends on <error detail>.

Appendix

Event Summary

The tables below lists log messages with log level, log text and short description of variable parts used in the event text.

Event Category ERROR

LOG LEVEL	EVENT TEXT / Description Variable Parts
10	failed to import name (major status <uint1> [<uint2>/<uint3>/<uint4>], minor status <uint5> [<uint6>/<uint7>/<uint8>])
	<uint1>: GSSAPI major status
	<uint2>: Value of highest byte of GSSAPI major status
	<uint3>: Value of second highest byte of GSSAPI major status
	<uint4>: GSSAPI major status
	<uint5>: GSSAPI minor status
	<uint6>: Value of highest byte of GSSAPI minor status
	<uint7>: Value of second highest byte of GSSAPI minor status
	<uint8>: Value of lowest 16Bit of GSSAPI minor status
10	failed to acquire service creadentials (major status <uint1> [<uint2>/<uint3>/<uint4>], minor status <uint5> [<uint6>/<uint7>/<uint8>])
	<uint1>: GSSAPI major status
	<uint2>: Value of highest byte of GSSAPI major status
	<uint3>: Value of second highest byte of GSSAPI major status
	<uint4>: GSSAPI major status
	<uint5>: GSSAPI minor status
	<uint6>: Value of highest byte of GSSAPI minor status
	<uint7>: Value of second highest byte of GSSAPI minor status
	<uint8>: Value of lowest 16Bit of GSSAPI minor status
10	<str1>: GSS calls completed with errors (major status <uint1> [<uint2>/<uint3>/<uint4>], minor status <uint5> [<uint6>/<uint7>/<uint8>])
	<str1>: Session Name
	<uint1>: GSSAPI major status
	<uint2>: Value of highest byte of GSSAPI major status

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<uint3>: Value of second highest byte of GSSAPI major status
	<uint4>: GSSAPI major status
	<uint5>: GSSAPI minor status
	<uint6>: Highest byte of minor status
	<uint7>: Value of second highest byte of GSSAPI minor status
	<uint8>: Value of lowest 16Bit of GSSAPI minor status
10	<str1>: Error (GSS_C_GSS_CODE): <str2>
	<str1>: Session Name
	<str2>: GSSAPI error description for major status
10	<str1>: Error (GSS_C_MECH_CODE): <str2>
	<str1>: Session Name
	<str2>: GSSAPI error description for minor status
10	<str1>: received invalid request code
	<str1>: Session Name
10	<str1>: received invalid request: <str2>
	<str1>: Session Name
	<str2>: Exception text
10	<str1>: received invalid request: unknown exception
	<str1>: Session Name
10	Failed to obtain credentials for host service. Check your Kerberos installation.
10	GSS Error (major status): <uint1> [<uint2>/<uint3>/<uint4>] (<str1>)
	<uint1>: GSSAPI major status
	<uint2>: Value of highest byte of GSSAPI major status
	<uint3>: Value of second highest byte of GSSAPI major status
	<uint4>: GSSAPI major status
	<str1>: GSSAPI error description for major status
10	Kerberos Error (minor status): <uint1> [<uint2>/<uint3>/<uint4>] (<str1>)
	<uint1>: GSSAPI minor status
	<uint2>: Highest byte of minor status
	<uint3>: Value of second highest byte of GSSAPI minor status
	<uint4>: Value of lowest 16Bit of GSSAPI minor status
	<str1>: GSSAPI error description for minor status
10	Value '<chr1>' for GUARDIANATTRIBUTESEPARATOR not acceptable, using default '<chr2>'.
	<chr1>: Separator
	<chr2>: Comma
10	Value <str1> for SFTPEDITLINEMODE not a supported value.
	<str1>: Value configured for parameter SFTPEDITLINEMODE
10	Value <int1> for SFTPEDITTABSIZE not acceptable, <str1>.
	<int1>: Number of spaces replacing a TAB
	<str1>: Error description
10	Value <str1> for SFTPEXCLUSIONMODEREAD not a supported value.

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Value configured for parameter SFTPEXCLUSIONMODEREAD
10	Value <str1> for SFTPEXCLUSIONMODEWRITE not a supported value.
	<str1>: Value configured for parameter SFTPEXCLUSIONMODEWRITE
10	request code <int1>
	<int1>: Request Code
10	APILIB error <int1>
	<int1>: Error
10	SFTPSERV serving <str1>@<str2> is stopping, reason: <str3>.
	<str1>: User name
	<str2>: Remote host TCP/IP address
	<str3>: Reason
10	could not change to user's SFTP-INITIAL-DIRECTORY '<str1>', chdir failed with error <int1>
	<str1>: Initial SFTP directory as configured for an SSH user
	<int1>: Error number
10	could not lock user into SFTP-INITIAL-DIRECTORY '<str1>', chroot failed with error <int1>
	<str1>: Initial SFTP directory as configured for an SSH user
	<int1>: Error number
10	Value '<str1>' for RECORDDELIMITER not acceptable, <str2>.
	<str1>: End of record indicator
	<str2>: Error description
10	Value <int1> for SFTPEDITLINESTARTDECIMALINCR not in allowed range.
	<int1>: Value configured for parameter SFTPEDITLINESTARTDECIMALINCR
10	Value <int1> for SFTPEDITLINENUMBERDECIMALINCR not in allowed range.
	<int1>: Value configured for parameter SFTPEDITLINENUMBERDECIMALINCR
10	Functionality is restricted to HP internal usage
10	Please contact License.Manager@hp.com for a full license
10	No valid license found: functionality is restricted to HP internal usage
10	Could not listen on interface <str1>, port <int1>: <str2>
	<str1>: Interface the SSH2 process listens on
	<int1>: Port
	<str2>: Exception text
10	Retrying to listen in <int1> second<str1>
	<int1>: Retry listen time in seconds
	<str1>: Plural s
10	Exception occurred: <str1>
	<str1>: Exception text
10	Retrying to listen
10	<str1>: Failure during decoding of Kerberos5 OID received in <str2> authentication request for user '<str3>', <uint1> decode errors
	<str1>: Session Name
	<str2>: Authentication method name
	<str3>: User name

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<uint1>: Decode error number
10	<str1>: could not add HPSIM key: <str2>
	<str1>: Session Name
	<str2>: Exception text
10	Invalid runmode. SSH2 terminating.
10	Valid runmodes are CLIENT, DAEMON, SERVER (same as DAEMON), ADMIN, NOADMIN, CLIENT_ADMIN, SERVER_ADMIN, DAEMON_ADMIN or ALL.
10	Failed to create private host key file <str1>
	<str1>: Private key file name
10	Failed to write private host key to file <str1>
	<str1>: Private key file name
10	Error loading private host key: <str1>. Possible mismatch of CUSTOMER setting between file creation and file access.
	<str1>: Exception text
10	Connection timed out.
10	Unexpected exception during initialization: <str1>
	<str1>: Exception text
10	Unexpected exception in main wait loop: <str1>
	<str1>: Exception text
10	<str1>: could not impersonate user <str2>, error <int1>
	<str1>: Session name
	<str2>: System user name
	<int1>: Error
10	<str1>: user is mapped to a SAFEGUARD ALIAS
	<str1>: Session name
10	<str1>: If SAFEGUARD is configured with PASSWORD-REQUIRED, start SSH2 with SAFEGUARD-PASSWORD-REQUIRED TRUE
	<str1>: Session name
10	<str1>: failed to create passive data connection tunnel from <str2> to <str3> (<str4>)
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
	<str4>: Description
10	Invalid state <int1> in FtpTunnelLayer::Notify, closing channel
	<int1>: State

Event Category WARNING

LOG LEVEL	EVENT TEXT / Description Variable Parts
20	gssapi kex failed: <str1>
	<str1>: Error message
20	<str1>: GSS KEX disabled: <str2>
	<str1>: Session Name
	<str2>: Error text
20	<str1>: forwarding remote <str2> connection from <str3> to <str4> failed (<str5>)
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
	<str5>: Description
20	<str1>: listen request from remote failed, could not listen on <str2> (<str3>)
	<str1>: Session Name
	<str2>: Normalized address and port to bind
	<str3>: Error text
20	<str1>: listen on <str2> terminated with error: <str3>
	<str1>: Session Name
	<str2>: Address and port to listen on
	<str3>: Error text
20	<str1>: forwarding <str2> connection from <str3> to <str4> failed (<str5>)
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
	<str5>: Description
20	<str1>: forwarding <str2> connection from <str3> (accepted on <str4>) to remote failed (<str5>)
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
	<str5>: Description
20	<str1>: request from user <str2> rejected: <str3>
	<str1>: Session Name
	<str2>: Guardian user name
	<str3>: TCP/IP ModeText
20	<str1>: request rejected: <str2>
	<str1>: Session Name
	<str2>: Text
20	<str1>: session rejected: NonStop SSH not licensed for general usage.

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
20	<str1>: SSH client access denied: SSH2 not licensed for general usage.
	<str1>: Session Name
20	<str1>: could not add KNOWNHOST <str2> to database for local system user <str3>: <str4>
	<str1>: Session Name
	<str2>: Known host
	<str3>: Owner of new knownhost record
	<str4>: Exception text
20	<str1>: update of stored password <str2> for local system user <str3> failed. password is frozen
	<str1>: Session Name
	<str2>: Name of password record stored in SSH2 database
	<str3>: Owner of password record
20	<str1>: could not add or update stored password <str2> for local system user <str3>: <str4>
	<str1>: Session Name
	<str2>: Name of password record stored in SSH2 database
	<str3>: Owner of password record
	<str4>: Exception text
20	<str1>: Unexpected WRITEREAD from SSH client
	<str1>: Session Name
20	<str1>: Unexpected READ from SSH client
	<str1>: Session Name
20	<str1>: Unexpected WRITE from SSH client
	<str1>: Session Name
20	<str1>: cannot forward data because remote side has closed the channel, ignoring data
	<str1>: Session Name
20	<str1>: client access to known host <str2> denied, known host entry (known by local system user <str3>) is frozen
	<str1>: Session Name
	<str2>: Known host
	<str3>: Owner of known host entry
20	<str1>: client access to known host <str2> (known by local system user <str3>) denied, public remote host key received is different to stored one
	<str1>: Session Name
	<str2>: Known host
	<str3>: Owner of known host entry
20	<str1>: client access to unknown host at <str2>, prompting local system user <str3> to continue.
	<str1>: Session Name
	<str2>: Normalized target host address and port
	<str3>: Login name
20	<str1>: client access to unknown host at <str2> denied. Local system user: <str3>
	<str1>: Session Name
	<str2>: Normalized target host address and port

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str3>: Login name
20	<str1>: exception during host verification (local system user <str2>): <str3>
	<str1>: Session Name
	<str2>: Login name
	<str3>: Exception text
20	<str1>: Authentication of <str2> succeeded
	<str1>: Session Name
	<str2>: User name
20	<str1>: Authentication failed
	<str1>: Session Name
20	<str1>: gssapi authentication failed: <str2>
	<str1>: Session Name
	<str2>: Error messageError message
20	<str1>: request rejected: Forwarding error - USER <str2> is not permitted to open port <int1> on host <str3>.
	<str1>: Session Name
	<str2>: Name of USER record
	<int1>: Forwarding destination port
	<str3>: Normalized forwarding destination host address
20	<str1>: request rejected: Forwarding error - USER <str2> is not permitted to listen on port <int1> on host <str3>.
	<str1>: Session Name
	<str2>: Name of USER record
	<int1>: Source port
	<str3>: Normalized local host address
20	<str1>: failed to open channel, reason: <str2>
	<str1>: Session Name
	<str2>: Description
20	<str1>: channel request failed
	<str1>: Session Name
20	<str1>: error on channel: <str2>
	<str1>: Session Name
	<str2>: Exception text
20	<str1>: Remote Forwarding Error: <str2>
	<str1>: Session Name
	<str2>: Error text
20	<str1>: error on ssh session: <str2>
	<str1>: Session Name
	<str2>: Exception text
20	<str1>: aborting SSH session, reason: <str2>
	<str1>: Session Name
	<str2>: Reason
20	<str1>: forwarding from <str2> to <str3> denied, SSH2 parameter <str4> set to false

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
	<str4>: ALLOWTCPFORWARDING
20	<str1>: forwarding from <str2> to <str3> denied, USER <str4> not found in database and PARAM <str5> set to true
	<str1>: Session Name
	<str2>: Normalized originator host address and por
	<str3>: Normalized target host address and port
	<str4>: Guardian user name
	<str5>: RESTRICTIONCHECKFAILEDDEFAULT
20	<str1>: forwarding from <str2> to <str3> denied, RESTRICTION-PROFILE PERMIT-OPEN for USER <str4> does not include target host/port
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
	<str4>: User name
20	<str1>: forwarding from <str2> to <str3> denied, USER <str4> not permitted to initiate TCP forwarding
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
	<str4>: Guardian user name
20	<str1>: forwarding from <str2> to <str3> denied, RESTRICTION-PROFILE FORWARD-FROM for USER <str4> does not include originator host
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
	<str4>: User name
20	<str1>: listen request on <str2> denied, SSH2 parameter <str3> set to false
	<str1>: Session Name
	<str2>: Normalized address and port to bind
	<str3>: ALLOWTCPFORWARDING
20	<str1>: listen request on <str2> denied, USER <str3> not found in database and PARAM <str4> set to true
	<str1>: Session Name
	<str2>: Normalized address and port to bind
	<str3>: Guardian user name
	<str4>: RESTRICTIONCHECKFAILEDDEFAULT
20	<str1>: listen request on <str2> denied, USER <str3> not permitted to initiate TCP forwarding
	<str1>: Session Name
	<str2>: Normalized address and port to bind
	<str3>: User name
20	<str1>: listen request on <str2> denied, RESTRICTION-PROFILE PERMIT-LISTEN for USER <str3> does not include local address/port

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
	<str2>: Normalized address and port to bind
	<str3>: User name
20	<str1>: forwarding from <str2> denied, USER <str3> not found in database and PARAM <str4> set to true
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Guardian user name
	<str4>: RESTRICTIONCHECKFAILEDDEFAULT
20	<str1>: forwarding from <str2> denied, RESTRICTION-PROFILE FORWARD-FROM for USER <str3> does not include originator host
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: User name
20	gssapi authentication failed: <str1>
	<str1>: Error message
20	Insane thread started
20	Insane Thread Count down <int1>
	<int1>: Counter value
20	Insane Thread was killed.
20	DEFINE <str1> was set to <<str2>>
	<str1>: Define name
	<str2>: File name
20	parameter SUBNET was evaluated
20	TCP/IP process is <str1>
	<str1>: Subnet Name
20	<str1>: remote <str2> forwarding request failed, server could not listen on <str3> (<str4>)
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized remote address and port
	<str4>: Description
20	<str1>: Error: <str2>.
	<str1>: Session Name
	<str2>: Exception text
20	<str1>: Disconnect from remote: <str2>
	<str1>: Session Name
	<str2>: Reason for disconnect
20	<str1>: User auth method mismatch, available: <str2>, requested <str3>
	<str1>: Session Name
	<str2>: Remaining authentication methods
	<str3>: Requested authentication method
20	<str1>: request rejected: authentication requested from host <str2> with unknown SSH user name <str3> (and <str4> is set to FALSE).

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
	<str2>: Remote host TCP/IP address
	<str3>: User name
	<str4>: AUTOADDSYSTEMUSERS
20	<str1>: request rejected: USER <str2> is not permitted to connect from host <str3> due to RESTRICTION-PROFILE settings.
	<str1>: Session Name
	<str2>: User name
	<str3>: Remote host TCP/IP address
20	<str1>: request rejected: USER <str2> is not permitted to connect from host <str3> due to ALLOW-MULTIPLE-REMOTE-HOSTS being false and user has already connected from <str4>.
	<str1>: Session Name
	<str2>: User name
	<str3>: Remote host TCP/IP address
	<str4>: Remote IP address of user session
20	<str1>: request rejected: USER <str2> is not permitted to connect because the configured SYSTEM-USER <str3> is frozen (and SSH2 parameter <str4> is set to false).
	<str1>: Session Name
	<str2>: User name
	<str3>: System user name
	<str4>: ALLOWFROZENSYSTEMUSER
20	<str1>: Authentication denied: SSH2 not licensed for general usage.
	<str1>: Session Name
20	<str1>: <str2> authentication for user '<str3>' not allowed
	<str1>: Session Name
	<str2>: Last authentication method tried
	<str3>: User name
20	<str1>: Authentication of user '<str2>' with method '<str3>' failed: <str4>
	<str1>: Session Name
	<str2>: User name
	<str3>: Authentication method name
	<str4>: Exception text
20	<str1>: <str2> authentication for user '<str3>' not supported, SYSTEM-USER: <str4>
	<str1>: Session Name
	<str2>: Authentication method name
	<str3>: User name
	<str4>: System user name
20	<str1>: Authentication of user '<str2>' failed: <str3>
	<str1>: Session Name
	<str2>: User name
	<str3>: Exception textError messageReason
20	<str1>: public key authentication failed, algorithm not supported

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
20	<str1>: public key authentication failed, too many keys
	<str1>: Session Name
20	<str1>: public key authentication failed, invalid signature
	<str1>: Session Name
20	<str1>: <str2> authentication failed: GSSAPI not available
	<str1>: Session Name
	<str2>: Authentication method name
20	<str1>: <str2> authentication failed: no GSS context established during key exchange
	<str1>: Session Name
	<str2>: Authentication method name
20	<str1>: <str2> authentication for user '<str3>' not supported
	<str1>: Session Name
	<str2>: Authentication method name
	<str3>: User name
20	<str1>: No more authentication requests possible for <str2>
	<str1>: Session Name
	<str2>: User name
20	<str1>: channel request for subsystem sftp denied
	<str1>: Session Name
20	<str1>: channel request for subsystem sftp rejected, sftp is not licensed
	<str1>: Session Name
20	<str1>: channel request for subsystem sftp denied (due to the SSH user's sftp security settings)
	<str1>: Session Name
20	<str1>: channel request for subsystem sftp denied (due to the SSH user's allowed subsystems settings)
	<str1>: Session Name
20	<str1>: channel request for subsystem sftp denied (due to the SSH2 process' allowed subsystem settings)
	<str1>: Session Name
20	<str1>: request for subsystem tacl rejected, not licensed
	<str1>: Session Name
20	<str1>: channel request for subsystem tacl denied (due to the SSH user's allowed subsystems settings)
	<str1>: Session Name
20	<str1>: channel request for subsystem tacl denied (due to the SSH2 process' allowed subsystem settings)
	<str1>: Session Name
20	<str1>: request for subsystem <str2> failed, invalid parameter <str3>
	<str1>: Session Name
	<str2>: Subsystem name
	<str3>: Text
20	<str1>: request for subsystem <str2> failed, invalid parameters
	<str1>: Session Name
	<str2>: Subsystem name

LOG LEVEL	EVENT TEXT / Description Variable Parts
20	<str1>: shell request from 6530 client rejected, not licensed
	<str1>: Session Name
20	<str1>: channel shell for 6530 command interpreter denied (due to the SSH user's ALLOW-CI settings)
	<str1>: Session Name
20	<str1>: shell request from 6530 client rejected, configured system user unknown
	<str1>: Session Name
20	<str1>: <str2> request rejected, shell access not licensed
	<str1>: Session Name
	<str2>: Request type
20	<str1>: <str2> request rejected, shell access denied
	<str1>: Session Name
	<str2>: Request type
20	<str1>: <str2> request rejected, configured system user unknown
	<str1>: Session Name
	<str2>: Request type
20	<str1>: <str2> process initialisation failed, could not chdir or chroot to user's SFTP-INITIAL-DIRECTORY, error <int1>
	<str1>: Session Name
	<str2>: Program
	<int1>: Error detail
20	<str1>: <str2> process initialisation failed, error <int1> during startup procedure
	<str1>: Session Name
	<str2>: Program
	<int1>: Error detail
20	<str1>: could not launch program <str2>, error <int1>, detail <int2>
	<str1>: Session Name
	<str2>: Program
	<int1>: Error
	<int2>: Error detail
20	<str1>: could not spawn program <str2>, error <int1>
	<str1>: Session Name
	<str2>: Program name of spawned process
	<int1>: Error
20	<str1>: pty request denied: pseudo terminal access not licensed (authentication dummy pty: <str2>)
	<str1>: Session Name
	<str2>: Pseudo terminal name used for authentication
20	<str1>: pty request denied: pseudo terminal access not licensed
	<str1>: Session Name
20	<str1>: pty request denied: pseudo terminal access not allowed for user <str2> (authentication dummy pty: <str3>)
	<str1>: Session Name
	<str2>: User name

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str3>: Pseudo terminal name used for authentication
20	<str1>: pty request denied: pseudo terminal access not allowed for user <str2>
	<str1>: Session Name
	<str2>: User name
20	<str1>: Could not allocate PTY: <str2> (authentication dummy pty: <str3>)
	<str1>: Session Name
	<str2>: Exception text
	<str3>: Pseudo terminal name used for authentication
20	<str1>: Could not allocate PTY: <str2>
	<str1>: Session Name
	<str2>: Exception text
20	<str1>: forwarding from <str2> to <str3> denied, port forwarding not licensed
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
20	<str1>: forwarding from <str2> to <str3> denied, ALLOWTCPFORWARDING or ALLOW-TCP-FORWARDING for USER <str4> is FALSE
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
	<str4>: User name
20	<str1>: forwarding from <str2> to <str3> denied, only port 21 (target) or 20 (originator) allowed for FTP
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
20	<str1>: listen request on <str2> denied, port forwarding not licensed
	<str1>: Session Name
	<str2>: Normalized address and port to bind
20	<str1>: forwarding from <str2> denied, only port 20 (originator) allowed for FTP data connections
	<str1>: Session Name
	<str2>: Normalized originator host address and port
20	<str1>: request rejected: user '<str2>' is not mapped to a SYSTEM-USER.
	<str1>: Session Name
	<str2>: User name
20	<str1>: session rejected: SSH2 not licensed for general usage.
	<str1>: Session Name
20	Expected IPv6 address for parameter <str1> because IP mode is <str2> but found TCP/IPv4 address <str3>. Using value <str4> instead.
	<str1>: Parameter name
	<str2>: TCP/IP mode
	<str3>: Value configured for parameter
	<str4>: Normalized interface address value

LOG LEVEL	EVENT TEXT / Description Variable Parts
20	Expected IPv6 address for parameter <str1> because IP mode is <str2> but found IPv4 address <str3>. Using value <str4> instead.
	<str1>: Parameter name
	<str2>: TCP/IP mode
	<str3>: Value configured for parameter
	<str4>: Normalized interface address value
20	Expected IPv4 address for parameter <str1> because IP mode is <str2> but found IPv6 address <str3>. Using value <str4> instead.
	<str1>: Parameter name
	<str2>: TCP/IP mode
	<str3>: Value configured for parameter
	<str4>: Normalized interface address value
20	Expected IPv4 address for parameter <str1> because IP mode is <str2> but found IPv4-compatible IPv6 address <str3>. Using value <str4> instead.
	<str1>: Parameter name
	<str2>: TCP/IP mode
	<str3>: Value configured for parameter
	<str4>: Normalized interface address value
20	Expected IPv4 address for parameter <str1> because IP mode is <str2> but found IPv4-mapped IPv6 address <str3>. Using value <str4> instead.
	<str1>: Parameter name
	<str2>: TCP/IP mode
	<str3>: Value configured for parameter
	<str4>: Normalized interface address value
20	Parameter <str1>: value '<str2>' is not a valid CPU list: <str3>. Using default value ('<str4>') instead.
	<str1>: Parameter name
	<str2>: Configured value
	<str3>: Reason for CPU set being invalid
	<str4>: Default value
20	Setting file security on '<str1>' from <oct1> to <oct2> failed, error <int1>
	<str1>: SSH database file name
	<oct1>: Current file security
	<oct2>: Expected file security
	<int1>: Error
20	Disabling incorrectly configured DNS resolving. Please correct DNS resolver configuration if needed and restart SSH2
20	Invalid file name: <str1>
	<str1>: String
20	File name could not be resolved: <str1>
	<str1>: String
20	Callback function on abend could not be initialized!
20	Expected version string was not received or version info line too long
20	<str1>: failed to create active data connection tunnel from <str2> to <str3> (<str4>)

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
	<str2>: Normalized originator host address and port
	<str3>: Normalized target host address and port
	<str4>: Description
20	<str1>: SSH FTP Error '<str2>'
	<str1>: Session Name
	<str2>: Exception text
20	<str1>: socket error '<str2>', aborting session
	<str1>: Session Name
	<str2>: Exception text
20	<str1>: unexpected error '<str2>', aborting session
	<str1>: Session Name
	<str2>: Exception text
20	<str1>: unknown error, aborting session
	<str1>: Session Name
20	<str1>: could not find target SSH and FTP address in '<str2>'
	<str1>: Session Name
	<str2>: Received command
20	<str1>: received command '<str2>': not valid while not connected to an FTP server
	<str1>: Session Name
	<str2>: Text
20	<str1>: FTP logon failed, reporting login failure to FTP client
	<str1>: Session Name
20	<str1>: connection to SSH server at <str2> failed, reporting failure to client
	<str1>: Session Name
	<str2>: Normalized target host address and port
20	<str1>: SSH user authentication failed, disconnecting.
	<str1>: Session Name
20	<str1>: SSH user authentication o.k.
	<str1>: Session Name
20	<str1>: failed to create SSH tunnel to FTP server at <str2> (<str3>), disconnecting SSH session
	<str1>: Session Name
	<str2>: Normalized target host address and port
	<str3>: Description
20	Cannot forward data because remote side has closed the channel, ignoring data
20	Configuration error regarding parameter <str1>: <str2>
	<str1>: CLIENTMODEOWNERPOLICY
	<str2>: Error number
20	User <str1>: Error occurred while checking if system user <str2> is frozen. Assuming system user is <str3>
	<str1>: Name
	<str2>: System user name

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str3>: Value "frozen" or "thawed"
20	Deleting user sessions records (user <str1>) created by no longer existing SSH2 processes failed: <str2>
	<str1>: User name
	<str2>: Exception text
20	Updating sessions record for user '<str1>' failed: <str2>
	<str1>: User name
	<str2>: Exception text
20	Updating sessions record (removing port <int1>) for user '<str1>' failed: <str2>
	<int1>: Port
	<str1>: User name
	<str2>: Exception text
20	Deleting all user sessions records failed: <str1>
	<str1>: Exception text
20	Deleting sessions record for user '<str1>' (process <str2>) failed: <str3>
	<str1>: User name
	<str2>: Process name
	<str3>: Exception text

Event Category INFO

LOG LEVEL	EVENT TEXT / Description Variable Parts
50	server credentials acquired successfully
50	<str1>: deleting credential cache '<str2>'
	<str1>: Session Name
	<str2>: Kerberos credentials cache file name
50	<str1>: GSS calls completed sucessfully
	<str1>: Session Name
50	<str1>: No system user name supplied, user credential cache will not be created
	<str1>: Session Name
50	No system user name supplied, user credential cache will not be created
50	<str1>: processing GSSAUTH_INIT_SECURITY_CONTEXT_REQUEST for user '<str2>'
	<str1>: Session Name
	<str2>: User initiating GSSAPI authentication
50	<str1>: processing GSSAUTH_ACCEPT_SECURITY_CONTEXT_REQUEST
	<str1>: Session Name
50	<str1>: security context was fully accepted for principal '<str2>'
	<str1>: Session Name
	<str2>: Client principal name
50	<str1>: processing GSSAUTH_VERFY_MIC_REQUEST

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
50	<str1>: caching credentials for user '<str2>'
	<str1>: Session Name
	<str2>: User initiating GSSAPI authentication
50	<str1>: credentials cache file name is '<str2>'
	<str1>: Session Name
	<str2>: Kerberos credentials cache file name
50	<str1>: processing GSSAUTH_GET_MIC_REQUEST
	<str1>: Session Name
50	<str1>: GSSAPI interface opened
	<str1>: Session Name
50	<str1>: GSSAPI interface closed
	<str1>: Session Name
50	<str1>: Exception in GSSAUTHContextService::OnWriteRead, returning error 22
	<str1>: Session Name
50	SFTPOSS version <str1> starting
	<str1>: SSH2 version
50	<str1>: forwarding remote <str2> connection from <str3> to <str4>
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
50	<str1>: closed forwarded remote <str2> connection from <str3> to <str4> <str5>
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
	<str5>: Reason
50	<str1>: remote <str2> forwarding request o.k., server listens on <str3>, forwarding to <str4>
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Remote address and port
	<str4>: Normalized target host address and port
50	<str1>: remote <str2> forwarding canceled, server listen on <str3> terminated
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Remote address and port
50	<str1>: forwarding request o.k., listening on <str2>
	<str1>: Session Name
	<str2>: Normalized address and port to bind
50	<str1>: cancel forwarding request, listening on <str2> terminated <str3>

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
	<str2>: Normalized address and port to bind
	<str3>: Reason
50	<str1>: forwarding <str2> connection from <str3> to <str4>
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
50	<str1>: forwarding <str2> connection from <str3> (accepted on <str4>) to remote
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
50	<str1>: closed forwarded <str2> connection from <str3> to <str4> <str5>
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
	<str5>: Reason
50	<str1>: closed forwarded <str2> connection from <str3> (accepted on <str4>) <str5>
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Normalized originator host address and port
	<str4>: Normalized target host address and port
	<str5>: Reason
50	<str1>: client session opened
	<str1>: Session Name
10	Please contact License.Manager@hp.com for a full license.
50	<str1>: added host as KNOWNHOST <str2> to database upon user request.
	<str1>: Session Name
	<str2>: Known host
50	<str1>: local system user <str2> aborted connection to unknown host, disconnecting because remote host key not verified.
	<str1>: Session Name
	<str2>: Login name
50	<str1>: connection failed, error <str2>
	<str1>: Session Name
	<str2>: Exception text
50	<str1>: client session closed, disconnecting from server
	<str1>: Session Name
50	<str1>: client session closed
	<str1>: Session Name

LOG LEVEL	EVENT TEXT / Description Variable Parts
50	<str1>: client access to known host <str2> (known by<str3><str4>)
	<str1>: Session Name
	<str2>: Known host
	<str3>: Local system user or ALL
	<str4>: Owner
50	<str1>: automatically updated KNOWNHOST <str2> via GSS key exchange (known by local system user <str3>)
	<str1>: Session Name
	<str2>: Known host
	<str3>: Owner of known host entry
50	<str1>: automatically accepted KNOWNHOST <str2> via GSS key exchange (entry known by <str3>)
	<str1>: Session Name
	<str2>: Known host
	<str3>: Owner of new knownhost record
50	<str1>: added unknown host identification as FROZEN HOST to database:<str2>
	<str1>: Session Name
	<str2>: Known host
40	<str1>: SSH client session established.
	<str1>: Session Name
50	<str1>: establishing remote <str2> port forwarding for <str3>.
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Target host name and port
50	<str1>: establishing local <str2> port forwarding for <str3>.
	<str1>: Session Name
	<str2>: Protocol
	<str3>: Target host name and port
40	<str1>: Port forwarding error: <str2>.
	<str1>: Session Name
	<str2>: Exception text
50	<str1>: requesting a pseudo terminal
	<str1>: Session Name
50	<str1>: sending subsystem request for subsytstem 'sftp'
	<str1>: Session Name
50	<str1>: sending shell request
	<str1>: Session Name
50	<str1>: sending exec request for command '<str2>'
	<str1>: Session Name
	<str2>: EXEC request command
50	<str1>: remote process terminated with exit code <int1>
	<str1>: Session Name
	<int1>: Exit status

LOG LEVEL	EVENT TEXT / Description Variable Parts
50	<str1>: channel request ok
	<str1>: Session Name
50	<str1>: server version string: <str2>
	<str1>: Session Name
	<str2>: SSH server software version
50	<str1>: session disconnected by server: <str2>
	<str1>: Session Name
	<str2>: Reason for disconnect
10	DEFINE =TCPIP^PROCESS^NAME has value '<str1>'
	<str1>: TCP/IP process name define
10	parameter SUBNET will be ignored and the define value will be used
50	<str1>: spawned program <str2> successfully (pid <int1>)
	<str1>: Session Name
	<str2>: Program name of spawned process
	<int1>: Process id of spawned process
50	<str1>: spawned program <str2> terminated with exit code <int1>
	<str1>: Session Name
	<str2>: Program name of spawned process
	<int1>: Completion code of spawned process
50	<str1>: launched program <str2> successfully (<str3>)
	<str1>: Session Name
	<str2>: Program name of launched process
	<str3>: Name of launched process
50	<str1>: launched program <str2> terminated with completion code <int1>
	<str1>: Session Name
	<str2>: Program name of launched process
	<int1>: Completion code of launched process
40	<str1>: SSH session established.
	<str1>: Session Name
50	<str1>: Sending banner message
	<str1>: Session Name
50	<str1>: Received 'Disconnect By Application' from remote: <str2>
	<str1>: Session Name
	<str2>: Reason for disconnect
40	<str1>: SSH session terminated
	<str1>: Session Name
10	SSH2 Server listening on interface <str1>, port <int1>
	<str1>: Interface the SSH2 process listens on
	<int1>: Port
50	<str1>: accepted connection from client
	<str1>: Session Name

LOG LEVEL	EVENT TEXT / Description Variable Parts
50	<str1>: auditing initiated.
	<str1>: Process name
50	<str1>: user '<str2>' automatically added to SSHCTL upon first authentication request using default user '<str3>'
	<str1>: Session Name
	<str2>: User name
	<str3>: User name
50	<str1>: user '<str2>' automatically added to SSHCTL upon first authentication request
	<str1>: Session Name
	<str2>: User name
40	<str1>: signature ok, authentication of <str2> successful
	<str1>: Session Name
	<str2>: User name
40	<str1>: accepting user '<str2>' without authentication
	<str1>: Session Name
	<str2>: User name
40	<str1>: Making user '<str2>' change the password
	<str1>: Session Name
	<str2>: User name
40	<str1>: password <str2> for user '<str3>', <str4> authentication successful
	<str1>: Session Name
	<str2>: Text "changed" if password was changed; else text "verified"
	<str3>: User name
	<str4>: Last authentication method tried
40	<str1>: gssapi authenticated principal is '<str2>'
	<str1>: Session Name
	<str2>: Client principal name
40	<str1>: principal '<str2>' mapped to local user '<str3>' (system user '<str4>')
	<str1>: Session Name
	<str2>: Client principal name
	<str3>: User name
	<str4>: System user name
40	<str1>: gssapi mic ok, authentication of '<str2>' successful
	<str1>: Session Name
	<str2>: User name
50	<str1>: channel request for subsystem sftp, launching sftp server
	<str1>: Session Name
50	<str1>: client version string: <str2>
	<str1>: Session Name
	<str2>: SSH client software version
50	<str1>: channel request for subsystem <str2>, launching <str3>
	<str1>: Session Name

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str2>: Subsystem name
	<str3>: Program
50	<str1>: channel request for 6530 shell, connecting to <str2>
	<str1>: Session Name
	<str2>: Program
50	<str1>: channel request for 6530 shell, launching <str2>
	<str1>: Session Name
	<str2>: Program
50	<str1>: channel request for 6530 shell, connecting to PTYSERVER <str2> <str3>
	<str1>: Session Name
	<str2>: Pseudo terminal server
	<str3>: Service name
50	<str1>: channel request for shell, connecting to <str2>
	<str1>: Session Name
	<str2>: Shell program
50	<str1>: channel exec request, launching <str2> -c <str3>
	<str1>: Session Name
	<str2>: Shell program
	<str3>: Command to execute
50	<str1>: channel shell request, launching <str2>
	<str1>: Session Name
	<str2>: Command to execute
50	<str1>: channel request for shell, connecting to PTYSERVER <str2> <str3>
	<str1>: Session Name
	<str2>: Pseudo terminal server
	<str3>: Service name
50	<str1>: Allocated PTY <str2> (authentication dummy pty: <str3>)
	<str1>: Session Name
	<str2>: Pseudo terminal name
	<str3>: Pseudo terminal name used for authentication
50	<str1>: Allocated PTY <str2>
	<str1>: Session Name
	<str2>: Pseudo terminal name
50	<str1>: routing connection to target ftp port <int1>
	<str1>: Session Name
	<int1>: Target port
10	No valid license found: restricting functionality to HP internal usage
10	CRYPTOPP version <str1>
	<str1>: Crypto++ library version
10	Invalid value specified for parameter <str1>: <str2>. Using default value <str3>.
	<str1>: ALLOWINFOSSH2

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str2>: Parameter value
	<str3>: Default value of ALLOWINFOSSH2
10	SSH config database file <str1> does not exist, creating.
	<str1>: SSH database file name
10	SSH config database <str1> opened.
	<str1>: SSH database file name
10	Initializing SSH2 ADMIN run mode.
10	Initializing SSH2 CLIENT run mode.
10	Initializing SSH2 DAEMON run mode.
10	Loading private key from <str1>
	<str1>: Private key file name
10	Private key file <str1> does not exist, creating <int1> bits key.
	<str1>: Private key file name
	<int1>: Number of host key bits
30	Host key algorithm: <str1>
	<str1>: Host key algorithm
30	Host key MD5 fingerprint: <str1>
	<str1>: MD5 finger print
30	Host key Bubble-Babble: <str1>
	<str1>: Key bubble babble
50	<str1>: connected SSH tunnel to FTP server at <str2>
	<str1>: Session Name
	<str2>: Normalized target host address and port
50	Accepted connection from <str1>, port <int1>, sessionid is <str2>
	<str1>: Normalized originator host address
	<int1>: Tunnel originator port
	<str2>: Session Name
50	<str1>: connection closed by FTP client
	<str1>: Session Name
50	<str1>: connection closed by FTP server, closing SSH session
	<str1>: Session Name
50	<str1>: user '<str2>' connects via SSH host at <str3> to FTP server on port <str4>
	<str1>: Session Name
	<str2>: User name
	<str3>: Normalized target host address and port
	<str4>: Normalized FTP target host and address
40	<str1>: received password from FTP client, sending SSH authentication request, method none
	<str1>: Session Name
40	<str1>: received quit command from FTP client
	<str1>: Session Name
40	<str1>: received FTP server welcome, attempting to login with SSH credentials

LOG LEVEL	EVENT TEXT / Description Variable Parts
	<str1>: Session Name
40	<str1>: received password request, sending user password
	<str1>: Session Name
40	<str1>: FTP logon o.k, reporting success to FTP client
	<str1>: Session Name
30	<str1>: connected to SSH server at <str2>
	<str1>: Session Name
	<str2>: Normalized target host address and port
30	<str1>: SSH server version is <str2>
	<str1>: Session Name
	<str2>: Server version
30	<str1>: Host key MD5 is <str2>
	<str1>: Session Name
	<str2>: Host key MD5 value
30	<str1>: Host key bubble-babble is <str2>
	<str1>: Session Name
	<str2>: SSH server bubble babble
40	<str1>: SSH authentication with method none failed, sending SSH authentication request, method password
	<str1>: Session Name
40	<str1>: initiating SSH tunnel to FTP server at <str2>
	<str1>: Session Name
	<str2>: Normalized FTP target host and address
30	SSH2 FTP over SSH gateway listening on interface <str1>, port <int1>
	<str1>: TCP/IP network interface
	<int1>: Port
50	Warning: channel data exception <str1>
	<str1>: Exception text
50	Warning: unknown channel data exception
50	Warning: error: <str1>
	<str1>: Exception text

Copyright Statements

As explained in the "SSH Protocol Reference" chapter, SSH2 uses some open source code for some components. This section of the appendix contains the various copyright notes.

All patent rights of the various contributors to the open source components of SSH2 are acknowledged.

OpenSSL Copyright Statement

The OpenSSL toolkit is licensed under a dual-license (the OpenSSL license and the original SSLeay license). See the license text below.

OpenSSL License

Copyright (c) 1998-2000 The OpenSSL Project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

All advertising materials mentioning features or use of this software must display the following acknowledgment: This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org)

The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org

Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

SSLeay license

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved. This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL. This library is free for commercial and non-commercial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)". The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).

If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.

OpenSSH Copyright Statement

This file is part of the OpenSSH software.

The licences which components of this software fall under are as

follows. First, we will summarize and say that all components

are under a BSD licence, or a licence more free than that.

OpenSSH contains no GPL code.

1)

```
* Copyright (c) 1995 Tatu Ylonen <ylo@cs.hut.fi>, Espoo, Finland
*                    All rights reserved
*
* As far as I am concerned, the code I have written for this software
```

```
 * can be used freely for any purpose.  Any derived versions of this
 * software must be clearly marked as such, and if the derived work is
 * incompatible with the protocol description in the RFC file, it must be
 * called by a name other than "ssh" or "Secure Shell".

[Tatu continues]
 *  However, I am not implying to give any licenses to any patents or
 * copyrights held by third parties, and the software includes parts that
 * are not under my direct control.  As far as I know, all included
 * source code is used in accordance with the relevant license agreements
 * and can be used freely for any purpose (the GNU license being the most
 * restrictive); see below for details.

[However, none of that term is relevant at this point in time.  All of
these restrictively licenced software components which he talks about
have been removed from OpenSSH, i.e.,

 - RSA is no longer included, found in the OpenSSL library
 - IDEA is no longer included, its use is deprecated
 - DES is now external, in the OpenSSL library
 - GMP is no longer used, and instead we call BN code from OpenSSL
 - Zlib is now external, in a library
 - The make-ssh-known-hosts script is no longer included
 - TSS has been removed
 - MD5 is now external, in the OpenSSL library
 - RC4 support has been replaced with ARC4 support from OpenSSL
 - Blowfish is now external, in the OpenSSL library

[The licence continues]

Note that any information and cryptographic algorithms used in this
software are publicly available on the Internet and at any major
bookstore, scientific library, and patent office worldwide.  More
information can be found e.g. at "http://www.cs.hut.fi/crypto".

The legal status of this program is some combination of all these
permissions and restrictions.  Use only at your own responsibility.
You will be responsible for any legal consequences yourself; I am not
making any claims whether possessing or using this is legal or not in
your country, and I am not taking any responsibility on your behalf.


                        NO WARRANTY
```

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

2)

The 32-bit CRC compensation attack detector in deattack.c was contributed by CORE SDI S.A. under a BSD-style license.

```
 * Cryptographic attack detector for ssh - source code
 *
 * Copyright (c) 1998 CORE SDI S.A., Buenos Aires, Argentina.
 *
 * All rights reserved. Redistribution and use in source and binary
 * forms, with or without modification, are permitted provided that
 * this copyright notice is retained.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
 * WARRANTIES ARE DISCLAIMED. IN NO EVENT SHALL CORE SDI S.A. BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY OR
 * CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR MISUSE OF THIS
 * SOFTWARE.
 *
 * Ariel Futoransky <futo@core-sdi.com>
 * <http://www.core-sdi.com>
```

3)

ssh-keyscan was contributed by David Mazieres under a BSD-style license.

```
 * Copyright 1995, 1996 by David Mazieres <dm@lcs.mit.edu>.
 *
 * Modification and redistribution in source and binary forms is
 * permitted provided that due credit is given to the author and the
 * OpenBSD project by leaving this copyright notice intact.
```

4)

The Rijndael implementation by Vincent Rijmen, Antoon Bosselaers
and Paulo Barreto is in the public domain and distributed
with the following license:

```
 * @version 3.0 (December 2000)
 *
 * Optimised ANSI C code for the Rijndael cipher (now AES)
 *
 * @author Vincent Rijmen <vincent.rijmen@esat.kuleuven.ac.be>
 * @author Antoon Bosselaers <antoon.bosselaers@esat.kuleuven.ac.be>
 * @author Paulo Barreto <paulo.barreto@terra.com.br>
 *
 * This code is hereby placed in the public domain.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHORS ''AS IS'' AND ANY EXPRESS
 * OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
 * BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5)

One component of the ssh source code is under a 3-clause BSD license,
held by the University of California, since we pulled these parts from
original Berkeley code.

```
 * Copyright (c) 1983, 1990, 1992, 1993, 1995
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
```

```
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
```

6)

```
Remaining components of the software are provided under a standard
2-term BSD licence with the following names as copyright holders:

    Markus Friedl
    Theo de Raadt
    Niels Provos
    Dug Song
    Aaron Campbell
    Damien Miller
    Kevin Steves
    Daniel Kouril
    Wesley Griffin
    Per Allansson
    Nils Nordman
    Simon Wilkinson

 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
```