H363L USER-Access®

for HP x86 L-Series NonStop Systems

Release 3.4.2

Memo to Users

October 2019

6450 Wedgwood Road N Ste 103
Maple Grove, MN 55311

Document Revision Record

Revision	Description
MTU-H363L-R3.4.1	First revision of document associated with H363L Release 3.4.1
MTU-H363L-R3.4.2	Corrected Restore call, minor fixes.

Introduction

This document contains information on release 3.4.2 of the Network Executive Software H363L USER-Access product for HP x86 L-Series NonStop systems including product installation instructions.

This document should be reviewed in its entirety before the product is installed.

Prerequisites

Prerequisites for this product are:

- HP Integrity NonStop host processor running NonStop OS release L16.05 or above.
- One or more Ethernet LAN interfaces.

Related Publications

The following book provides more information about this software product:

H363 USER-Access® for HP NonStop Systems

Software Support Limitations

Modifications to H363L that are not specifically authorized by NetEx Software are prohibited.

Any unauthorized modifications to H363L may affect its operation and/or obstruct NetEx Software's ability to diagnose problems and provide corrections. Any work resulting from unauthorized modifications shall be paid by the customer at NetEx Software's then-current support rates and may result in the immediate termination of warranty/support coverage.

Service Notes

- The –CREATE REPLACE qualifier does not work correctly when the source and destination files are the same (i.e., the source and destination volume. subvolume and file name are exactly alike). If a SEND or RECEIVE command is issued this way, the source/destination file will be empty.
- The parameters SI_USERNAME and SI_PASSWORD should be defined in the SICONFIG file as part of the standard install. Failure to define a valid non-privileged username/password can result in a potential security loophole. Valid SI_USERNAME and SI_PASSWORD parameters allow the Service Initiator to reset itself back to a non-privileged user ID following each successful login. It also allows the Service Initiator to be stopped (e.g., STOP $SI) by the user designated as SI_USERNAME. This could be a standard OPERATOR account but should not be SUPER.SUPER or group.SUPER (a group manager). Any UID of *,255 should be avoided.

  If SI_USERNAME or SI_PASSWORD is missing or invalid, appropriate warning messages will be displayed in the Service Initiator log file (SILOG). However, the Service Initiator will continue to operate.

  The SI_PASSWORD can be encrypted in the SICONFIG file. To do so, use the ENCRYPT string function (or ENCRYPT alias) provided with USER-Access. Refer to the section titled “Password Encryption” for more information on password encryption.
- Due to USER-Access enhancements to support NonStop EXPAND networks, previous support for 5-character process names (e.g., $USERA) may no longer be valid. Limit any USER-Access client and server process names to a maximum of 4 characters (plus the dollar sign).
- Three or more consecutive failures to logon through the Service Initiator (via the USER-Access “CONNECT” or “LOGIN” command) will cause the Service Initiator to be suspended for 60 seconds before returning the message ‘Invalid username or password (SI363-8002)’. This delay will be experienced by the user currently connecting to the HP NonStop host. Other connect attempts during this interval will return a “service not offered” message. This is a result of the VERIFYUSER system service.
- HP NonStop local/remote exit status needs more explanation.. The command is assumed to complete successfully (status = STOP). Then each line of the output stream is scanned for the string ‘ABENDED:’ as the first eight characters. If this string is detected, the command is considered to have failed (status = ABEND). The ON commands for LOCAL_ERROR and REMOTE_ERROR will respond to this command failure.

  In order for USER-Access to determine exit status from a TACL command, USER-Access now also recognizes the string ‘TACLEXIT:nnn’. If this string is detected, the exit status becomes ‘nnn’.
- File transfer modes BACKUP, RESTORE and COPY do not capture/restore the file characteristics during peer-to-peer transfers unless they are used with BACKUP/RESTORE. BACKUP, RESTORE, and COPY modes do not save file characteristics outside of the NonStop BACKUP utility.
- Execution of the first local/remote NonStop command will experience a delay while activating the command interpreter. This same delay is experienced following a previous command abort (due to a keyboard interrupt) or when the command interpreter is changed (e.g., SET LOCAL/REMOTE COMINT) since a new command interpreter is being activated.
- It is possible to write a NonStop USER-Access script that contains a tight loop during which keyboard interrupts are ignored. On tight loops (e.g., LOOP: GOTO LOOP) keyboard interrupts have no effect.

Installation

This section provides complete installation procedures for the H363L x86 L-Series NonStop USER-Access product.

Before you Begin

- Please review all of the information in the "Service Notes" section before proceeding with the installation.
- If your system is currently running a previous release of USER-Access, you can preserve it by renaming the existing USER-Access root subvolume. This allows easy recovery if the current installation fails. You may also wish to copy some of the site files from the previous release once the current release is installed. Use FUP to rename all the files in the current subvolume.

  For example, if the current release is installed in $SYSTEM.UA, rename all of the files to the subvolume $SYSTEM.UASAV1 as follows:

  ```
  15> FUP RENAME $SYSTEM.UA.*, $SYSTEM.UASAV1.*
  ```

  Once the new release is installed, site control files (e.g., SCLIENT) can be copied from the renamed subvolume to the new USER-Access root subvolume.
- The installation procedure assumes your logon command interpreter is a named TACL process. If this is not the case, logon to a named TACL process before continuing.
- The USER-Access release is loaded into a distribution volume.subvolume and is then installed into the target volume.subvolume. Existing files in the target volume.subvolume are replaced with the new release files. Any user-modified control files are automatically preserved.

  Nothing is modified in the distribution volume.subvolume during the installation process allowing repeated installations into different target volume.subvolumes, each with different install options. To preserve disk space the distribution volume.subvolume can be deleted once USER-Access is fully installed into the target volume.subvolume.

  The distribution volume.subvolume requires less than 2000 pages (4 Mbytes) of disk space. The target volume.subvolume requires less than 2500 pages (5 Mbytes) of disk space.
- The installation procedures and examples assume a distribution volume.subvolume of "$SYSTEM.UA342" and a target volume.subvolume of $SYSTEM.UA. However, any valid volume.subvolume (with adequate space) can be selected. If you choose an alternate volume.subvolume, be sure to replace $SYSTEM.UA342 or $SYSTEM.UA with the alternate volume.subvolume in subsequent commands.
- TCP port number 6900 is used by the USER-Access Service Initiator as its default listen port. This is the recommended port number for use by USER-Access. However, if other applications require this port number, see "Appendix A: Updating the TCP/IP Network Control Files" for instructions to change these defaults.

Installation Procedure

To perform a complete installation and verification of this product, follow the steps in this section.

Step 1. Select USER-Access volume.subvolume

Set your current volume to volume.subvolume that you will copy the USER-Access distribution file to:

```
16> VOLUME $SYSTEM.UA342
```

Step 2. Download the Distribution

From either the NetEx Software download service (contact Support) or the distribution CD, perform a binary file transfer of UA342PAK to the distribution volume.subvolume you selected in the previous step.

Step 3. UNPAK the Distribution

Issue the following UNPAK command:

```
17> UNPAK UA342PAK,*.*.*,LISTALL,MYID,TAPEDATE,VOL <$vol.subvol>
```

The list of restored files should agree with the file list in the Distribution Contents section.

Step 4. Install the Software from the Distribution

The USER-Access INSTALL procedure is a TACL routine that prompts for the site install options and invokes another TACL routine named MAKE with the selected parameters.

The Installation Process

Invoke the USER-Access INSTALL procedure using the prompts shown below. Default responses are enclosed in square brackets:

```
$SYSTEM.UA342 6> run install

The target volume.subvol defines the location where USER-Access
is installed.

Enter the target volume.subvol [$SYSTEM.UA]?

Starting USER-Access install
Copying .. UAMACROS
FILES DUPLICATED:  1
Copying .. SISTART
FILES DUPLICATED:  1
Copying .. SICONFIG
FILES DUPLICATED:  1
Copying .. SHOWLOG
FILES DUPLICATED:  1
Copying .. UASRVR
FILES DUPLICATED:  1
FILES DUPLICATED:  1
```

```
FILES DUPLICATED:  1
FILES DUPLICATED:  1
FILES DUPLICATED:  1
FILES DUPLICATED:  1
FILES DUPLICATED:  1
FILES DUPLICATED:  1
FILES DUPLICATED:  1
#output Linking.....
Linking.....
eld /in LINKIN2  , out UALINK.LINKIN2  /
eld /in LINKIN3  , out UALINK.LINKIN3  /
eld /in LINKIN4  , out UALINK.LINKIN4  /
eld /in LINKIN5  , out UALINK.LINKIN5  /
FILES DUPLICATED:  1
FILES DUPLICATED:  1
FILES DUPLICATED:  1
FILES DUPLICATED:  1
Finished USER-Access install
$SYSTEM.UA 7>
```

Be sure to scan the generated output for any errors or warning messages.

Note: During the installation procedure (between "Starting …" and "Finished …") the install program does the following:

- Copies and edits the USER-Access control files, substituting the target volume.subvolume for the USER-Access root directory. Any user modified control files (e.g., SCLIENT, SSERVER, and so on) are preserved with an appropriate message displayed.
- Generates a ROOT object module defining the target volume.subvolume as the USER-Access root directory.
- Links the USER-Access modules: CLIENT, SERVER, SVCINIT, and CONTROL.
- Sets the file security on the USER-Access control files and executable images.

Step 5. Start the Service Initiator

The Service Initiator is a program that services USER-Access CONNECT or LOGIN requests from hosts on the network. It runs as a NOWAIT process named **$SI**. To use an alternate process name (e.g., running multiple Service Initiators) replace **$SI** with a valid process name in the following commands.

Set your current volume to the USER-Access target volume.subvolume:

```
19> VOLUME $SYSTEM.UA
```

You must make any site-specific changes to the **SICONFIG** file before starting the Service Initiator. Refer to the section on "Service Initiator Keywords" for more details. Be sure to define the parameters **SI_USERNAME** and **SI_PASSWORD**. Remove the comment character (#) from the sample lines and change the group.userid and password entries to that of a valid non-privileged account. This account should not be SUPER.SUPER or group.SUPER (i.e., the UID should not be *,255). Make any other changes appropriate for your site:

```
20> EDIT SICONFIG
      .
      .
```

To activate a new Service Initiator, issue the following command:

```
21> RUN SISTART $SI
```

To verify that the Service Initiator is running, the named process $SI should display when the following command is issued:

```
22> STATUS $SI
```

Once the Service Initiator has started successfully, users on remote hosts can connect into the NonStop host by providing a valid username and password. The Service Initiator will activate a server process with a home terminal of $SI. A site manager can monitor the amount of USER-Access activity by listing the active processes associated with $SI (i.e., STATUS *, TERM $SI).

Step 6. Verify USER-Access

A simple verification command procedure has been supplied as part of the USER-Access release. This verification procedure prompts you for the name of the local host and a valid username/password on this host (for verification purposes, the username/password specified should at least have the privileges you do - specify your username/password to be safe). It attempts to connect back to the host, logon, and send and receive several files.

To run the verification command, first load the **UAMACROS** file to define the TACL macro USER (used to invoke the USER-Access CLIENT) along with other USER-Access macros:

```
24> LOAD UAMACROS
```

Then invoke the USER-Access client with the VERIFY script:

```
25> USER VERIFY
```

There should be several messages printed to your terminal followed by a final “Verification Successful” message. If an error is encountered during the verification, the procedure will terminate and an error message will be printed.

This completes the installation of USER-Access on your client host.

Making USER-Access Available To Other Users

In order to make USER-Access easily available to other NonStop users, load the TACL macros found in the file **$SYSTEM.UA.UAMACROS** from your site startup files (TACLLOCL or TACLBASE) or from your individual user startup file (TACLCSTM). This allows users logging into TACL to invoke the USER-Access client by simply typing “USER”. If “USER” conflicts with other site definitions, the recommended alternative is “USERA”.

To automatically start the Service Initiator following a system RELOAD, run the “$SYSTEM.UA.SISTART” command as part of the reload process. Make sure any network processes have been started first.

Update Summary

Release 3.4.2

This release is functionally compatible with the 3.4.2 release of H363I which incorporates the corrected use of Restore.

#8507: Correct create line to entry-sequenced
#8481: FILE_GETRECEIVEINFO_ maximum reply count is unsigned
#8482: Help is not available for 'logins' alias
#3483: ENCRypt alias forces username to uppercase
#8484: LOGINS alias does not expand username in encrypt function

Release *3.4.1*

This is a new product release, but functionally compatible with the 3.4.1 release of H363I.

Appendix A: Updating the TCP/IP Network Control Files

The network control file **$SYSTEM.ZTCPIP.SERVICES** can be updated to change the USER-Access default.

You must first select a unique port number. Port numbers in the range 0-1023 are reserved for TCP network services. USER-Access uses port number 6900 as the default. For example, add the following line to the end of the file:

```
user        6900/tcp    USER
```

Appendix B: Running with TCP/IP over Multiple Ethernet Adapters

This section describes how USER-Access can be configured on a NonStop system with multiple Ethernet adapters. This configuration is most often used to increase aggregate TCP/IP performance from a single system. In a configuration such as this, the NonStop host has a TCP/IP hostname for each Ethernet adapter, thus the two Ethernet adapters make the NonStop function as two TCP/IP hosts on the network. This section shows how to select an adapter when running USER-Access.

An example of a NonStop system is one that has two TCP/IP processes running, one for each Ethernet adapter installed. The first TCP/IP process is named $ZTC0 and is identified by TCP/IP hostname TAN0. The second TCP/IP process is named $ZTC1 and is identified by TCP/IP hostname TAN1.

The key to running USER-Access on a particular Ethernet adapter is to set the "**TCPIP^PROCESS^NAME**" environment variable to the proper value. This is done with the NonStop "**PARAM**" command. If this variable is not set to a particular TCP/IP process name prior to running USER-Access, the default process name will be used (the default is setup when NonStop TCP/IP is installed).

The rest of this section describes:

- Running the USER-Access TCP/IP Client over multiple Ethernet adapters.
- Running the USER-Access TCP/IP Service Initiator and Server over multiple Ethernet adapters.

Running the USER-Access TCP/IP Client

The USER-Access TCP/IP Client can be started such that it will run using either TCP/IP process. If a TCP/IP process name is not specified prior to invoking the client, the default TCP/IP process name is used.

For example, to run the USER-Access client on TCP/IP process name $ZTC0, issue the following:

```
TACL> add define =tcpip^process^name, file $ztc0
TACL> user
```

To run the client on TCP/IP process name $ZTC1, the following commands should be issued:

```
TACL> add define =tcpip^process^name, file $ztc1
TACL> user
```

The commands to run the client could be defined in a TACL macro to automatically start the client on a particular TCP/IP process name. For example, to define a TACL macro named USER0 to always start the client on TCP/IP process $ZTC0, put the following commands in your system or user TACL script:

```
?section user0 macro
add define =tcpip^process^name, file $ztc0
user %*%
```

Running the USER-Access TCP/IP Service Initiator and Server

In order to connect into this example NonStop system over either Ethernet adapter, two different service initiators must be started - one for each TCP/IP process. Then one can access this NonStop by connecting to hostname TAN0 or TAN1. Given two TCP/IP processes ($ZTC0 and $ZTC1), the following steps will

start two independent Service Initiators, one will offer service USER on $ZTC0 and the other will offer service USER on $ZTC1:

1. Start the editor and create a new file that will start both Service Initiators.

   TACL> *EDIT $system.ua.GOSI*

   The following log and configuration file names should not be qualified with a $volume.subvolume because they are created in $SYSTEM.UA (or where USER-Access was installed).

   Edit the following lines into the file to start the Service Initiators:

   ```
   ?TACL MACRO
   ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC0
   RUN $system.ua.SISTART $SI0 SILOG0 SICFG0
   ADD DEFINE =TCPIP^PROCESS^NAME, FILE $ZTC1
   RUN $system.ua.SISTART $SI1 SILOG1 SICFG1
   ```

2. Make a copy of the Service Initiator configuration file for $ZTC0.

   TACL> *FUP COPY $system.ua.SICONFIG $system.ua.SICFG0*

   Start the editor with the file $system.ua.SICFG0 and find the following line:

   ```
   USER   SERVER   run $system.ua.uasrvr /name/ -si -service %s
   ```

   Add parameters to the line to indicate the TCP/IP process name to use for the USER-Access server:

   ```
   USER   SERVER   run $system.ua.uasrvr -tcp $ZTC0 /name/ -si
   -service %s
   ```

3. Make a copy of the Service Initiator configuration file for $ZTC1.

   TACL> *FUP COPY $system.ua.SICONFIG $system.ua.SICFG1*

   Start the editor with the file $system.ua.SICFG1 and find the following line:

   ```
   USER   SERVER   run $system.ua.uasrvr /name/ -si -service %s
   ```

   Add parameters to the line to indicate the TCP/IP process name to use for the server:

   ```
   USER   SERVER   run $system.ua.uasrvr -tcp $ZTC1 /name/ -si
   -service %s
   ```

Appendix C: Stopping the Service Initiator

The Service Initiator can be stopped by using the Service Initiator Control program $SYSTEM.UA.CONTROL. To stop the Service Initiator, enter:

```
24> RUN $SYSTEM.UA.CONTROL STOP -H host
```

Substitute your local host name for ‘host’.

Appendix D: Service Initiator Keywords

The Service Initiator configuration file, **$SYSTEM.UA.SICONFIG**, contains a list of keywords that can be set to alter the way the Service Initiator operates for a given SERVICE being offered. Each keyword is listed here along with a brief description of the value that can be assigned to it:

SI_USERNAME

The group.userid of a non-privileged account used to reset the Service Initiator back to a non-privileged state following a successful logon. The username should not be SUPER.SUPER or group.SUPER (i.e., a UID of *,255).

SI_PASSWORD

The password associated with the account specified for the SI_USERNAME parameter. The SI_PASSWORD value can be encrypted. To do so, use the ENCRYPT string function (or ENCRYPT alias) provided with USER-Access. Refer to the "Password Encryption" section of this document for more information on password encryption.

SERVER

Specifies the command that is used to invoke or run the USER-Access server. The '%s' that appears in this string marks the command line position where the service name to offer is passed (e.g., USER001).

COMMAND

The command interpreter used for logon and activating the USER-Access server. Note that this command interpreter has no relationship to the command interpreter used for local or remote command execution. TACL is the default.

CPU

A round-robin list of CPU numbers for use by the USER-Access servers. By default, all USER-Access servers are started on the same CPU as the Service Initiator. The server load can be balanced by providing a list of CPUs on which to execute (e.g., CPU 0 1 2 3). A single CPU can be favored by repeating it in the list (e.g., CPU 0 1 1 2 3). The Service Initiator CPU is represented by the value -1 (e.g., CPU -1 4 alternates between the Service Initiator CPU and CPU #4).

Note: On systems where the $CMON process makes CPU assignments this parameter will have no effect.

USERNAME

Specifies the guest username if none is provided on the connect request.

PASSWORD

Specifies the guest password if none is provided on the connect request.

OPERATOR

Specifies a password that is required when issuing commands through the CONTROL program (e.g., for shutting down the Service Initiator).

LOGTIMEOUT

Specifies the logon timeout in seconds. This is used to terminate a logon request that for some unknown reason, is hanging around.

MINIMUM

Specifies the minimum session number that will be offered for this service. For example, a MINIMUM value of 5 would result in SERVICE “USER” being offered as “USER005” up to MAXIMUM (below).

MAXIMUM

Specifies the maximum session number that will be offered for this service. A value of 30, for example would, result in the last offer of “USER” being “USER030”, before the offers started over at MINIMUM.

TRACE

Allow different levels of tracing for the Service Initiator. Refer to the comments in the configuration file for a description of the different trace levels.

VERBOSE

Specifies whether the LOGON output is displayed back to the connecting user. The LOGON output is the normal TACL startup information that displays when a user logs onto a NonStop system.

Appendix E: Internal Features

Password Encryption

The string function ENCRYPT has been added to USER-Access. The purpose of this function is to encrypt host passwords which later will be used by USER-Access to establish host connections. This approach eliminates the security risk of having readable (clear-text) passwords stored in files.

Format:

```
encrypt(password, [username])
```

Where:

password

Specifies the password you want to encrypt. The encrypted form of this password is returned by the ENCRYPT string function. The encrypted form can be stored in script files containing USER-Access CONNECT commands.

username

Optionally specifies the username associated with the local USER-Access process that will issue the CONNECT command. This username is used as a secondary encryption key for the specified password. When USER-Access is later run it queries the operating system for the username running the current process. USER-Access then uses this username as one of its keys in decrypting the password. The value for username must be entered in uppercase to match the username value returned by the NonStop operating system. A value of '*' (single asterisk) tells the USER-Access ENCRYPT function to use the current username running the USER-Access process as the secondary key. You must be running as the same user which will later run USER-Access to issue the CONNECT command.

For example, encrypt the password COBRA using the NonStop username SUPER.GEM1 as the local username for secondary encryption. Use the USER-Access TEXT command to display the encrypted results:

```
User> text {encrypt("COBRA", "SUPER.GEM1")}
User: *249eece8e4203b189
```

The ENCRYPT Alias

To simplify the use of the ENCRYPT string function, an ENCRYPT alias is provided in the SCLIENT startup file in the USER-Access distribution. The ENCRYPT alias definition is shown:

```
set alias ENCrypt {} {dfn(1, "goto skip")} !
     ask -secure -prompt "Enter password? " 1 !
     ask -prompt "Enter optional username (or '*')? " 2 !
     skip: set global pw {encrypt(1, upper(2))} !
     text The encrypted password is {pw}
```

For example, the ENCRYPT alias could be used to encrypt the same password COBRA with the same secondary key SUPER.GEM1 shown previously:

```
User> encrypt
Enter password? COBRA   (password does not display)
Enter optional username (or '*')? SUPER.GEM1
User: The encrypted password is *249eece8e4203b189
```

Note the following items regarding the ENCRYPT alias:

1. The password is prompted in -SECURE mode to avoid displaying on the terminal.
2. The ENCRYPT alias can be invoke with ‘password’' and optional ‘username’ passed as alias parameters to avoid prompting. However, the password will display.
3. The optional ‘username’ is forced to uppercase using the UPPER string function.
4. The resulting encrypted password is stored in a global variable PW for later reference.

Examples

Example 1: Encrypting Passwords Stored in a USER-Access Input Script File

Suppose a job running under the local NonStop username NSC.JONES inputs the USER-Access script $SYSTEM.SCRIPTS.MVS1 during program execution and the script $SYSTEM.SCRIPTS.MVS1 contains the following line:

```
CONNECT mvs1 admin7 secret
```

To avoid storing the password ‘secret’ in readable form in the script file, the password is encrypted by invoking the USER-Access client and using the ENCRYPT alias:

```
User> encrypt secret NSC.JONES
User: The encrypted password is *26f17e2a4c9c65c56
```

Username NSC.JONES is specified because that is the local NonStop username under which the USER-Access job that uses the connect/login information will run. Using a local text editor, modify the input script $SYSTEM.SCRIPTS.MVS1 to look like:

```
CONNECT mvs1 admin7 *26f17e2a4c9c65c56
```

Example 2: Using USER-Access to Generate the Input Script File

As you can see in the ENCRYPT alias definition, the global variable ‘pw’ is set to the encrypted password value. This value can be used to generate an input file containing the USER-Access CONNECT command to be later referenced by a USER-Access script. We can use the USER-Access OUTPUT command to generate the script file $SYSTEM.SCRIPTS.MVS1 to connect to the host ‘mvs1’ as user ‘admin7’ with the password ‘secret’ (as shown in example #1):

```
User> encrypt secret NSC.JONES
User: The encrypted password is *26f17e2a4c9c65c56
User> set output prefix
User> output $system.scripts.mvs1
User> text CONNECT mvs1 admin7 {pw}
User> output
```

The resulting file $SYSTEM.SCRIPTS.MVS1 will contain the following line:

```
CONNECT mvs1 admin7 *26f17e2a4c9c65c56
```

USER-Access Data Compression

Support has been added to USER-Access for data compression and expansion during file transfer. The new SEND/RECEIVE qualifiers are:

COMPRESS - compress the source data stream (on/off)

EXPAND - expand the destination data stream (on/off)

METHOD - the method of compression (RLE/LZW)

Supported compression methods are RLE and LZW methods.

The RLE compression method uses a simple Run Length Encoding algorithm that counts strings of repeated characters (usually spaces or nulls). The RLE method will never grow data that is already compressed (except for the addition of the compression header).

The LZW method uses the Lempel-Ziv-Welch algorithm for finding common substrings. This method is deterministic and can be performed on the fly. Block compression is performed with an adaptive reset whereby the code table is cleared when the compression ratio decreases. This method generally provides the best overall compression ratio, but requires significantly more CPU resource than the RLE method. LZW compression ratios for character data are typically 40% to 60% of the original data size. LZW compression ratios for binary data are difficult to predict. Applying LZW compression to already compressed data could actually increase the data size up to 130% of the original size.

The following examples demonstrate file transfers using the -COMPRESS and -EXPAND qualifiers.

Send a binary source file 'data' with data compression enabled. The destination file 'data.cmp' contains the compressed data:

```
User> send -mode stream data data.cmp -compress
```

Receive the same compressed file expanding the data stream back to the original binary file:

```
User> receive -mode stream data.cmp data -expand
```

The same binary data file can be compressed, sent across the network and expanded into the destination file:

```
User> send -mode stream data -compress data -expand
```

One-sided compress/expand (the first two examples) is possible when connected to earlier releases (pre-R10) of USER-Access for all supported modes except CHARACTER. Two-sided compress/expand (the last example) requires that both sides (client and server) support compression.

Only certain combinations of -COMPRESS and -EXPAND are valid with the various USER-Access transfer modes. The following table shows which combinations are valid (Yes) and which are not valid (No):

Transfer Mode	-COMPRESS only	-EXPAND only	Both -COMP/-EXP
CHARACTER	Yes	Yes	Yes
RECORD	No	No	No
STREAM	Yes	Yes	Yes
BACKUP	Yes	No	Yes

Transfer Mode	-COMPRESS only	-EXPAND only	Both -COMP/-EXP
RESTORE	No	Yes	Yes
COPY	No	No	Yes
V1CHAR	No	No	No

Character Mode Compression

Both sides (client and server) of a CHARACTER mode transfer must support compression (release R10 and later). In addition, when transferring between hosts with different native character sets (e.g., ASCII to EBCDIC) there are some subtle problems caused by the fact that only the USER-Access client performs code conversion.

The character set of the compressed data is stored in the compression header that prefixes the compressed data stream. This information can be used during expansion to determine if code conversion must be performed.

The following table illustrates the various combinations of CHARACTER mode compress/expand. The source and destination file types are shown as well as any code conversion issues:

USER-Access command	Source	Destination	Code Conversion performed…
send -compress	text	stream	by client before compress
receive -compress	text	stream	not done - server pushes an informative message - flags its native char set in header
send -expand	stream	text	not done - error if char set in header does not match server's char set
receive -expand	stream	text	by client after expand if char set in header does not match client's native char set
send -compress -expand	text	text	by client before compress
receive -compress -expand	text	text	by client after expand

Command Piping

Command piping is the mechanism used for NonStop BACKUP/RESTORE where a data stream is directed to/from the USER-Access (TAPE) device. In this case the USER-Access named process looks like a tape device to the BACKUP and RESTORE utilities.

This mechanism has been generalized to allow command piping for any application or NonStop utility that supports I/O to a named process file. The source or destination of a file transfer can be a 'piped command' where the data is piped directly to/from the application or NonStop utility without intermediate staging of data on disk. This usually provides a performance advantage as well as eliminating the disk space requirements.

Piped commands are flagged in the USER-Access SEND/RECEIVE command by prefixing the command string with an exclamation mark (!). Piped commands containing embedded blanks must be enclosed within double quotes. Piped commands usually contain a special tag string such as (TAPE) that gets replaced by USER-Access with a named process file. For example, if the USER-Access process name is $Z347, the tag string (TAPE) gets replaced with the named process file $Z347.#TAPE. Then USER-Access monitors its $RECEIVE queue looking for open/close system messages with the special process name qualifiers (e.g., #TAPE).

There are several tag strings recognized by USER-Access. They are:

(TAPE)

Piping mechanism used where the USER-Access named process looks like a tape device to the NonStop BACKUP and RESTORE utilities.

(PIPE)

General named process piping mechanism. Each logical record is processed individually.

(BLOCK)

Piping mechanism where records are blocked or unblocked using the FUP format described for the VARIN/VAROUT parameter. Performance is improved by eliminating the overhead of reading/writing individual logical records.

The following examples of command piping use the NonStop FUP utility to perform a variety of "filtering" operations. Recognize that FUP is used to demonstrate functionality. However any NonStop utility, TACL macro or user application can utilize the piped command facility:

Example #1: Send the local structured file LOCDATA to the remote file REMDATA while padding each record with blanks (ASCII 32) to 132 characters:

```
User> send "!fup copy locdata,(pipe),recout 132,pad 32" remdata
```

Example #2: Receive the remote file REMDATA to the local structured file LOCDATA while trimming trailing blanks:

```
User> receive remdata "!fup copy (pipe),locdata,trim 32"
```

Example #3: Load a key-sequenced file KEYDATA from a sorted remote file REMDATA:

```
User> receive remdata "!fup load (pipe),keydata,sorted"
```

Example #4: The example above can be optimized by having USER-Access block the records in VARIN format using the (BLOCK) tag string. By default, USER-Access uses 32000 byte blocks:

```
User> receive remdata "!fup load (block),keydata,sorted, -
More>> varin,blockin 32000"
```

Example #5: Copy 1000 records from the local keyed file KEYDATA starting at the record with primary key "SMITH". Block the data in VAROUT format. Send to the remote file REMDATA using record mode to preserve binary data. Notice the required set of quotes around SMITH to escape the embedded quotes:

```
User> send "!fup copy keydata,(block),first key ""SMITH"", -
More>> count 1000,varout,blockout 32000" remdata -mode record
```

Example #6: If no tag string is specified, then the normal OUTPUT (for a source command) or INPUT (for a destination command) is piped. The following will send the FUP INFO of all the files on my current volume to the remote file VOLINFO:

```
User> send "!fup info *.*" volinfo
```

Example #7: Piped commands can be initiated locally (as shown in the previous example) or remotely by any USER-Access client connected to a NonStop server. The same volume information could be received by a remote client as follows:

```
User> receive "!fup info *.*" volinfo
```

Running USER-Access in Slave Mode

In all of the previous examples, USER-Access has been operating as the "master" process controlling the "slave" piped commands. The command piping facility allows USER-Access to operate as a slave process interacting with external commands. Simply use the piped command syntax without any command - just the exclamation mark (!) followed by the appropriate tag string. This is a flag to USER-Access to read/write its named process file being accessed by an external command. USER-Access looks for an OPEN system message with the appropriate name qualifier (e.g., #TAPE, #PIPE or #BLOCK). In order to synchronize with the external command, USER-Access sends a command interpreter WAKEUP message (-20) to its creator. This could be TACL or any user application invoking USER-Access with the NEWPROCESS system service.

The following example uses a TACL macro to activate USER-Access as a "nowait" named process $USER. USER-Access commands are taken from the text file UASCRIPT (shown later). Once USER-Access is started, the TACL macro will PAUSE waiting for the WAKEUP message. Then a FUP COPY is directed to the USER-Access slave process:

```
?tacl macro
user /name $USER, in UASCRIPT, out UALOG, nowait/
pause
fup copy locdata, $USER.#PIPE
```

The USER-Access script file UASCRIPT contains the following:

```
connect host username password
send !(pipe) remdata
exit
```

The USER-Access output will appear in UALOG. Multiple SEND and RECEIVE commands can be processed. Each must be synchronized with a PAUSE command in the TACL macro.

Additional Piped Command Qualifiers

Some additional qualifiers can be included with the tag strings to fine tune the piped command processing. These qualifiers are:

Open Timer

A numeric value (in seconds) specifying how long to wait for the piped command to actually open the USER-Access named process. The default open timer is 5 minutes (300 seconds). You can increase or decrease the open timer based on your requirements. A value of zero disables the open timer (i.e., never times out).

NOWAIT

Boolean flag indicating that the piped command is being run in NOWAIT mode. USER-Access will continue processing even though the command interpreter indicates command completion. This flag is especially useful when invoking a piped command that "kicks off" another job that will eventually open the named pipe process. Completion of the "kick off" process does not terminate piped command processing.

NOWAKEup

Boolean flag that disables the WAKEUP message to the creator process. When USER-Access is operating as a "slave" process (described earlier) the WAKEUP messages is not always necessary to synchronize processing. This flag prevents the WAKEUP message from being sent.

The following examples expand on some of the previous examples. Once again the FUP utility is used to demonstrate functionality:

Example #1: Send the local file LOCDATA to the remote file REMDATA. Run the piped command in NOWAIT mode:

```
User> send "!fup /nowait/ copy locdata,(pipe,nowait)" remdata
```

Example #2: Run USER-Access as a "slave" process and send the piped data to the file REMDATA. Set an open timer of 30 seconds. Don't send the WAKEUP message to the creator:

```
User> send !(pipe,30,nowakeup) remdata
```