# eFT213

## NetEx/eFT™ for IBM z/OS™ Operating Systems

**Release 5.5**

**Software Reference Manual**

REF-EFT213-R5.5

# Revision Record

| Revision | Description |
|---|---|
| 01 (01/2001) | Manual modified for NESi ownership |
| 02 (01/2001) | Reformat using NESi-specific template revision 12/15/2000. |
| 03 (04/2005) | • Updated for conversion to eFT product.<br>• All documentation changes noted in Memo To Users from prior releases added.<br>• Addition of FTP alias commands.<br>• Other minor grammatical and formatting changes. |
| 04 (06/2006) | • Updated copyright year.<br>• Changed references of "//SYSIN DD" in JCL examples to "//STDIN DD".<br>• General format changes for better readability. |
| 05 (11/2009) | • Corrections for release 5.4<br>• Updates copyright year.<br>• ISPF Panels supported.<br>• 0Cx on very long passwords corrected<br>• 0Cx on very long variables corrected<br>• Command length issues documented<br>CLIST allocations of STDIN should be changed to include a blocksize equal to or greater than the screen size of the 3270 model that you are using. I.e.,<br>`ALLOC F(STDIN)  DS(*) REUSE BLKSIZE(4096) RECFM(U)`<br>• DEBUGON & DEBUGOFF aliases added<br>• DBGON & DBGOFF scripts added |
| 06 (08/2011) | Added missing eFT213 message "EFT213-2536" to message table. |
| 07 (11/2011) | • Corrections to show command output.<br>• Support for new licensing beginning in Release 5.4.4 |
| 08 (05/2012) | Additions for Release 5.4.5 |
| 09 (06/2018) | Minor corrections coinciding with Release 5.4.7 |
| R5.5 (5/2021) | New feature release for optional secure connections (IP only) |



You may submit comments on this document by sending e-mail to support@netex.com or by visiting our "Report a Problem" page on the web at http://www.netex.com/support/report-problem. Please make sure to include the title of the document in your problem report.

# Preface

This manual describes the eFT213 NetEx/eFT software for the IBM z/OS system. NetEx/eFT can be used in conjunction with Network Executive Software's NETwork EXecutive (NetEx) software or TCP/IP to allow the end-user to easily transfer files across the network.

This manual is intended for all users of NetEx/eFT and it contains all of the information necessary to expand the user's ability to the fullest extent of the software.

The manual is divided into seven parts plus one appendix:

"Introduction" gives a basic description of NetEx/eFT and a sample of a NetEx/eFT session.

"IBM z/OS Local User's Guide" describes the features of NetEx/eFT on z/OS systems as seen by the local user. This section includes a description of the commands in the local interface.

"IBM z/OS Remote User's Guide" describes features of eFT213 as seen by a remote user. This includes executing commands on a z/OS host remotely and transferring files to and from a remote z/OS host.

"File Handling Under IBM z/OS NetEx/eFT" describes the way IBM z/OS manipulates files. This includes examples of transferring files, transfer modes supported by the IBM z/OS NetEx/eFT, wildcard characters, and file specifications.

"Advanced Local User's Guide" describes the advanced features of NetEx/eFT on the IBM systems as seen by the local user.

"Command Descriptions," "FTP Alias Commands," "Host Independent Commands," and "General Alias Commands" provide detailed descriptions of all commands available in eFT213 NetEx/eFT.

# Reference Material

The following manuals contain related information.

| Number | Title and Description |
|---|---|
| MAN-REF-H210 | *H210IP NetEx/IP for Oz/OS Systems Software Reference Manual* |

# Notice to the Reader

The material contained in this publication is for informational purposes only and is subject to change without notice. Network Executive Software is not responsible for the improper use of any product options or the use of features not described in this publication. It assumes no responsibility for any errors that may appear in this publication. Refer to the revision record (at the beginning of this document) to determine the revision level of this publication.

Network Executive Software does not by publication of the descriptions and technical documentation contained herein, grant a license to make, have made, use, sell, sublicense, or lease any equipment or programs designed or constructed in accordance with this information.

This document may contain references to the trademarks of the following corporations:

# Corporation Trademarks and Products

| | |
|---|---|
| **Network Executive Software** | **NetEx®, USER-Access™, NetEx/eFT** |
| **International Business Machines Corp.** | **IBM, z/OS, VTAM** |

These references are made for informational purposes only.

The diagnostic tools and programs described in this manual are **not** part of the products described.

# Notice to the Customer

The installation information supplied in this document is intended for use by experienced System Programmers.

## Software Modification Policy

Modifications to eFT213 that are not specifically authorized by NetEx Software are prohibited.

Any unauthorized modifications to eFT213 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.

# Document Conventions

The following notational conventions are used in this document.

| Format | Description |
|---|---|
| **`displayed information`** | Information displayed on screen (or printed) is shown in **`this style`**. |
| *`user entry`* | *`This style`* is used to indicate information entered by the user. |
| **UPPERCASE** | The exact form of a keyword that is not case-sensitive or is issued in uppercase. |
| **MIXedcase** | The exact form of a keyword that is not case-sensitive or is issued in uppercase, with the minimum spelling shown in uppercase. |
| **bold** | The exact form of a keyword that is case-sensitive and all or part of it must be issued in lowercase. |
| *User-supplied name or string* | A user-supplied name or string. |
| [Brackets] | Keywords/parameters enclosed in brackets are optional. |
| No delimiter | Keywords/parameters without brackets are required. |
| value | Underlined parameters or option values are defaults. |
| <label> | The label of a key appearing on a keyboard. If "label" is in uppercase, it matches the label on the key (for example: <ENTER>). If "label" is in lowercase, it describes the label on the key (for example: <up-arrow>). |
| <key1><key2> | Two keys to be pressed simultaneously. |

# Glossary

**ASCII:** Acronym for American National Standard Code for Information Interchange.

**code conversion:** An optional feature in the NetEx interface that dynamically converts the host data from one character set to another.

**header:** A collection of control information transmitted at the beginning of a message, segment, datagram, packet, or block of data.

**host:** A data processing system that is connected to the network and with which devices on the network communicate. In the context of Internet Protocol (IP), a host is any addressable node on the network; an IP router has more than one host address.

**hostname:** A unique name of a host or server. The host name should be a text string consisting of the letters A through Z (upper or lower case), digits 0-9, minus sign (-), and the period (.). Note, the period is only allowed as the last character of the host name if it is the delimiter of the full domain name (FQDN). No spaces are permitted as part of a name. At least one character must be an alphabetic character and the last character must not be a minus sign or period. NetEx/IP hostnames are limited to 8 characters, while IP Hostnames are limited to 63 characters.

**Internet Protocol (IP):** A protocol suite operating within the Internet as defined by the *Requests For Comment* (RFC). This may also refer to the network layer (level 3) of this protocol stack (the layer concerned with routing datagrams from network to network).

**Network Configuration Table (NCT):** An internal data structure that is used by the NETEX configuration manager program to store all the information describing the network.

**NETwork EXecutive (NetEx):** A family of software designed to enable two or more application programs on heterogeneous host systems to communicate. NetEx is tailored to each supported operating system, but can communicate with any other supported NetEx, regardless of operating system.

NetEx can reside on the host.

NetEx is a registered trademark of Network Executive Software.

**Open Systems Interconnection (OSI):** A seven-layer protocol stack defining a model for communications among components (computers, devices, people, etc.) of a distributed network. OSI was defined by the ISO.

**path:** A route that can reach a specific host or group of devices.

**TCP/IP:** An acronym for Transmission Control Protocol/Internet Protocol. These communication protocols provide the mechanism for inter-network communications, especially on the Internet. The protocols are hardware independent. They are described and updated through *Requests For Comment (RFC)*. IP corresponds to the OSI network layer 3, TCP to layers 4 and 5.

**SSL:** An acronym for Secure Sockets Layer. SSL is a standard security protocol for establishing encrypted links between a web server and a browser in an online communication.

The use of SSL technology ensures all communication between the eFT server and client remains encrypted.

# Contents

# Figures



# Tables

# Introduction

## NetEx/eFT Overview

The Network Executive Software (NESi) NetEx/eFT software is a user interface to the NESi NETwork EXecutive (NetEx) software or standard TCP/IP. It provides the ordinary user with a means to move and manipulate files across a network using simple, easily remembered commands. In addition, NetEx/eFT provides extensive interactive help files so the user can become familiar with NetEx/eFT.

NetEx/eFT provides several advantages to network users. Among these are:

- **User-friendly** - Once NetEx/eFT is installed, you can transfer files and exercise other NetEx/eFT functions in very little time and with little training.
- **Tailorable** - The NetEx/eFT interface can be tailored to meet your needs at the host and the user levels. Default values can be set, aliases defined, etc., in site- and user-input files that are read by NetEx/eFT when it is invoked.
- **Common Interface** - The NetEx/eFT user interface is the same on all hosts. While the definition of a command may change from one host to another, the command remains the same to you.
- **Security** - NetEx/eFT uses the host computer's logon routines to provide security. You must be a valid user on both the local and the remote systems to access them. Some systems may allow a guest account, but this can be restricted by the security needs of the network.

## How NetEx/eFT Works

How NetEx/eFT functions is illustrated in Figure 1 on page 27. As the figure shows, the user (initiator) sends a request to the remote system (responder), including account and password information.

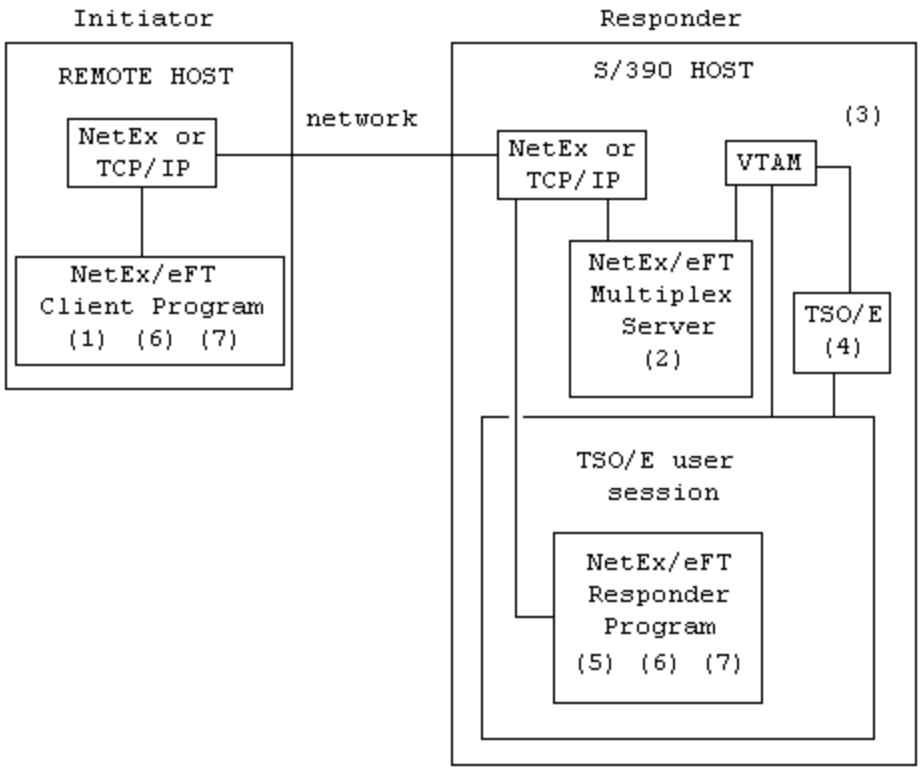



**Figure 1. Diagram of a NetEx/eFT Remote Connection Sequence**

1. Remote user invokes NetEx/eFT and issues the CONNECT command.
2. Initiator connects over the network to the Multiplex Server.
3. The Multiplex Server starts a VTAM session and issues a LOGON command to TSO/E.
4. TSO/E validates user ID and password and creates the TSO/E user session.
5. The NetEx/eFT Responder program is invoked by the Multiplex Server.
6. The Responder and CLIENT exchange terms of connection.
7. NetEx/eFT file transfers occur directly between the Initiator and Helper programs without involving the Multiplex server, VTAM, or TSO/E.

# Introduction to NetEx/eFT and z/OS

This manual describes the NetEx/eFT software for an IBM z/OS host running NetEx/IP or TCP/IP.

NetEx/eFT is a software product designed to simplify network communications. Network capabilities have been expanded to include the nontechnical user by reducing the user interface to a set of simple commands (**CONNECT**, **SEND**, **RECEIVE**, **DISCONNECT**, etc.).

# Sample IBM z/OS NetEx/eFT Session

This section gives a very brief example of a few of the functions that can be accomplished during a NetEx/eFT session. This sample session is meant to be only a simple introduction to NetEx/eFT and how it may appear to the local IBM z/OS user. The sections following this provide a more detailed look at the product and its features.

Users that have never seen NetEx/eFT may spend a couple of minutes following through this sample session. Users that are familiar with the product may skip directly to the next section.

To invoke NetEx/eFT, the EFT command is entered from the IBM z/OS TSO/E command line as:

```
%EFT
%EFTUSER - EFT starting (TCP)
 eFT:
 eFT: Welcome to MVS NTXeFT version 5.5.0-nnn N24
 eFT:
eFT>
```

The returning prompt in this sample session is **eFT>** although NetEx/eFT may be configured to return another prompt. The prompt informs the user that NetEx/eFT is waiting to accept a command.

A connection to any host in the network that is running NetEx/eFT can be made using the **LOGIN** alias. The **LOGIN** and **LOGON** aliases establish a connection with a zOS host named zos5. **LOGIN** prompts the user for login information such as remote username and password which it uses to establish a secure login on the remote host using the **CONNECT** command. The **LOGON** alias will also prompt for the account and profile parameters. The output returned is based on the host and username to which the connection is made. The connection is completed when a NetEx/eFT prompt appears. Notice that in this session, NetEx/eFT has been configured to prompt with the name of the remote host.

```
eFT> logon
 Hostname? yellowstone
 Username? test1
 Password?
 Account?
 Procedure?
 Qualifiers? -serv eft

13:15:42 MVS:Connected to Service Initiator on host 'yellowstone'.
========================================================================
Last login: Thu Jun  6 10:04:22 CDT 2019 from zpdt2-5.netexsw.com on pts/5
Directory: /mnt/home2/test1
Thu Jun  6 13:15:44 CDT 2019

========================================================================
13:15:47 MVS:Logged in as user 'test1'.
13:15:47 MVS:Connected to service '60137' on host 'yellowstone'.
13:15:47 MVS:Welcome to UNIX NETEX-eFT version 5.5.0-nnn N24
13:15:47 MVS:*
13:15:47 MVS:* Entered server.ua file
13:15:47 MVS:*
yellowstone>
```

If a connection fails, an error message is displayed. The error generally begins:

```
13:20:43 MVS:Failed to connect service 'EFT4' on host 'ZOSX' (UA-4105).
```

This is followed by either an appropriate network message or a remote system error message. If the host ZOSX is not in the network, for example, an error similar to the following would appear:

```
13:20:43 MVS:Host 'ZOSX' does not exist in configuration (UA-804).
13:20:43 MVS:Invalid TCP host name 'ZOSX' (EFT213-2501).
```

If the username/password combination was invalid, an error such as the one below would be seen:

```
eFT: Connected to Service Initiator on host 'yellowstone'.
 ========================================================================
 Login incorrect
 ========================================================================
 eFT: Failed to connect service 'eft' on host 'yellowstone' (UA-4105).
 eFT: Remote: Login failed (SI693-8011).
 eFT>
```

Since all logins are made through the security system of the remote host, the error message seen by the user will depend on the host to which the connection is being made.

Following a successful login as above, a **SHOW HOST** command can be used to display all remote host connections held by this NetEx/eFT session. Each session can support up to ten host connections. The command below reveals just one remote host connection. The connection displayed is the one just established by **LOGIN** at the beginning of this session.

```
yellowstone> show host
eFT:
eFT: active --> (1) Host=yellowstone Secure=OFF    User=test1
eFT:
yellowstone>
```

Once a connection is established, a **SHOW REMOTE** command can be issued to return useful information about the connection and the remote NetEx/eFT host. From the list below, for example, we see that the remote host character code is “EBCDIC”, the default directory/prefix is “TEST1”, and the NetEx/eFT version number is “eFT 5.5”.

```
yellowstone> show remote
eFT:
eFT: * BLOCKsize ......... 16384
eFT: * COPYRight ......... Copyright (C) 1999-2021, Network Executive Software,
Inc.
eFT:   DIRectory ......... /mnt/home2/test1
eFT: * GATEway ...........
eFT:   HOMEdir ........... /mnt/home2/test1
eFT: * HOST .............. yellowstone
eFT: * HOSTCODE .......... ASCII7
eFT: * HOSTTYPE .......... UNIX
eFT: * LicExp ............ 20191130
eFT: * LicKey ............ DNLJ-YAC2-AD7Q-CANW-7LBH-6UHP-VAJB-KFT5
eFT: * LicNotOper ........ 20200228
eFT: * LicProto .......... NTX IP SSL
eFT: * NODE ..............
eFT: * PID ............... 29920
eFT:   PREFix ............ Unix:
eFT: * PRODuct ........... EFT803
eFT:   QUIet ............. off
eFT: * ROOTdir ........... /usr/share/nesi/eftip/user
eFT: * SECure ............ off
eFT: * SERvice ........... 44890
eFT:   SHELL ............. /bin/bash
eFT: * SSLCipher .........
eFT: * SSLProto ..........
eFT: * SSLFIPSmode ....... off
eFT: * STATus ............
eFT: * TRANSlate ......... Network
```

```
eFT: * USERname .......... test1
eFT: * VERsion ........... 5.5
eFT:
eFT: * Informational qualifier (cannot be modified).
eFT:
yellowstone>
```

Similar information can also be displayed about the local IBM z/OS host by issuing the **SHOW LOCAL** command.

Once a connection is established to a remote host, users can issue commands to that host using the **REMOTE** command. The example below issues **REMOTE WHO**, a UNIX command which returns a list of users currently logged onto the remote UNIX system. Notice that a host-specific prefix appears in the left-hand column indicating the results are being returned from a host named "yellowstone".

```
yellowstone> remote who
Unix: root     pts/0        2019-04-02 14:52 (vnchost.netexsw.com)
Unix: root     pts/2        2019-05-09 09:38 (vnchost.netexsw.com)
Unix: root     pts/4        2019-05-09 09:39 (vnchost.netexsw.com)
Unix: test1    pts/5        2019-06-06 13:27 (zpdt2-5.netexsw.com)
yellowstone>
```

A listing of all remote UNIX files that reside in the remote user's account can be displayed using the **REMOTE** command followed by **DIRECTORY** (which says to display a directory file listing):

```
yellowstone> remote directory
Unix: total 4030660
Unix: drwxr-xr-t 35 test1 nesi    3477504 2019-05-30 14:40 .
Unix: drwxr-xr-x 61 root  root       4096 2017-01-19 11:17 ..
Unix: -rw-rw-r--  1 test1 nesi      48000 2019-11-30 15:21 abcde
Unix: -rw-rw-r--  1 test1 nesi         47 2019-06-18 16:32 badcmd1
Unix: -rw-rw-r--  1 test1 nesi        278 2019-12-04 13:11 bakrstfl
Unix: -rw--w----  1 test1 nesi       6538 2019-05-30 13:34 .bash_history
Unix: -rw-rw-r--  1 test1 nesi        191 2018-05-13 14:50 .bash_profile
Unix: -rw-rw-r--  1 test1 nesi       3478 2017-11-09 12:17 .bashrc
Unix: -rw-rw-r--  1 test1 nesi      11870 2019-08-14 09:37 betatst.datahuge
Unix: -rw-rw-r--  1 test1 nesi       2975 2019-11-24 16:10 'betatst.ua(en-
crypt1)'
Unix: -rw-rw-r--  1 test1 nesi       6195 2019-11-24 10:51 'betatst.ua(ftpa-
lias)'
Unix: -rw-rw-r--  1 test1 nesi       4422 2019-10-19 15:19 'betatst.ua(reccmp2)'
Unix: -rw-rw-r--  1 test1 nesi       2944 2019-11-19 15:06 'betatst.ua(setloc1)'
Unix: -rwxr-xr-x  1 test1 nesi       2964 2017-11-14 13:58 bfxtest
Unix: -rw-rw-r--  1 test1 nesi       8192 2019-12-08 11:13 binrec60
Unix: -rw-rw-r--  1 test1 nesi      48000 2019-10-08 17:01 binrec60.ua
Unix: drwx-w----  2 test1 nesi       4096 2016-06-23 01:23 bircd
Unix: -rw-r--r--  1 test1 nesi         18 2014-06-06 12:21 boa1
Unix: -rw-r--r--  1 test1 nesi         18 2014-06-06 11:02 boa2
Unix: -rw-r--r--  1 test1 nesi         18 2014-06-06 11:14 boa3
```

Besides issuing remote host commands, users can also issue local TSO/E commands from within a NetEx/eFT session. For example, to display a listing of all files or data sets on the local z/OS host with the user's TSO/E prefix, a **LOCAL LISTCAT** command can be issued. **LISTCAT** is the TSO/E command to display a directory listing:

```
zost> local listcat
 IN CATALOG:USERCAT1
 TEST1.ABC
 TEST1.ABCD
 TEST1.ABCDE
 TEST1.ASKDEF2
 TEST1.A1111
 TEST1.A2222
 TEST1.A3333
 TEST1.A4444
 TEST1.BACKFL
 TEST1.BACKUP
 TEST1.BAKRSTFL
 TEST1.BBB
 TEST1.BETATST.DATAA
 TEST1.BETATST.DATAB
 TEST1.BETATST.H210IPZ.DIST6885.DFILE
 TEST1.BETATST.H210IPZ.DIST6889.DFILE
 TEST1.BETATST.H210IPZ.DIST6894.DFILE
 TEST1.BINREC60
 TEST1.BLANKOFF
 TEST1.BLANKON
 TEST1.BLANKS
 TEST1.CHGDIR
 TEST1.CONUCHK
 TEST1.DAN1.S0W1.SPFLOG1.LIST
 TEST1.DAN1.S0W1.SPFTEMP0.CNTL
 TEST1.DAN3.S0W1.SPFLOG1.LIST
 TEST1.DATA
 TEST1.DATADD1
 TEST1.DATA5MR2
 TEST1.DATA5MR3
 TEST1.DAVEDAT1
 TEST1.DAVEDAT2
 TEST1.EFT-945
 TEST1.EFTLOG.NEW3
 TEST1.EFTLOG.NEW97
zost>
```

A major feature of NetEx/eFT is its implementation of a Host Independent Command set. Host independent commands, implemented as aliases, allow a user to issue similar commands on all hosts around the network, without having to learn each host's native command set. One of these host independent commands is **DIRECTORY**, as issued in the **REMOTE** command example to the UNIX host earlier. NetEx/eFT simply maps **DIRECTORY** to the equivalent host command for both the local and remote system. (Under UNIX, **DIRECTORY** maps to the UNIX command "**ls -al**". Under IBM z/OS, **DIRECTORY** translates to the TSO/E command **LISTCAT**). Now network users need only learn one network-wide command set. This command set can be the NetEx/eFT default one or one that the site defines (shown in detail later in the manual). Below is a second pass at a local directory listing, but this time using the host independent command **DIRECTORY**.

```
zost> local directory
IN CATALOG:USERCAT1
TEST1.ABC
TEST1.ABCD
TEST1.ABCDE
TEST1.ASKDEF2
TEST1.A1111
TEST1.A2222
TEST1.A3333
TEST1.A4444
TEST1.BACKFL
TEST1.BACKUP
TEST1.BAKRSTFL
TEST1.BBB
TEST1.BETATST.DATAA
TEST1.BETATST.DATAB
zost>
```

To transfer a file from the local host to the remote host, the **SEND** command is used. The example below sends the file or sequential data set "EARTH.UA" from the current local TSO prefix (or directory) "GUEST1" on the IBM z/OS host, to the current remote directory /home/guest on the UNIX host. Since all NetEx/eFT commands can be predefined with reasonable site defaults, the typical user would just type **SEND** followed by the source file name. The status line indicates the file has successfully been transferred. Notice that NetEx/eFT uses the source file name to create a default destination file name when one isn't specified.

```
zost> send earth.ua
eFT: Source                Destination                Size
eFT: --------------------  ------------------------  ------
eFT: GUEST1.EARTH.UA       /home/guest/earth.ua        750
zost>
```

With NetEx/eFT it is also very easy to transfer a group of files using a single command containing wildcarding. The example below sends all of the local sequential data sets having an ending qualifier of "UA" to the remote UNIX host, giving them an extension of ".UNIXUA". If any file transfer errors were encountered, they would be displayed in place of the status line below.

```
zost> send *.ua *.unixua
eFT: Source                Destination                      Size
eFT: --------------------  -------------------------------  ----
eFT: GUEST1.EARTH.UA       /home/guest/earth.unixua          750
eFT: GUEST1.OCEAN.UA       /home/guest/ocean.unixua          217
eFT: GUEST1.WORLD.UA       /home/guest/world.unixua           94
zost>
```

Wildcarding can also be used to select certain partitioned data set members to be transferred from IBM z/OS and stored as files on the remote host. The example below uses wildcarding to select all members that have "e" as the second character:

```
zost> send mvs.pds(?e*)
eFT: Source                Destination                    Size
eFT: --------------------  -----------------------------  ----
eFT: GUEST1.MVS.PDS(BETA)  /home/guest/beta.pds            215
zost>
```

A quick **REMOTE DIRECTORY** will act as a second verification that the files have indeed been transferred. Note the new files BETA.PDS, EARTH.UA, EARTH.UNIXUA, OCEAN.UNIXUA, and WORLD.UNIXUA below:

```
zost> remote directory
Unix: total 25506
Unix: drwxr-xr-x  10 guest   group1       1536 Jan 17 16:38 .
Unix: drwxr-xr-x  12 root    other         512 Jan 17 16:37 ..
Unix: -rw-------   1 guest   group1         71 Jan 17 16:37 .TTauthority
Unix: -rw-------   1 guest   group1        198 Jan 17 16:37 .Xauthority
Unix: -rw-r--r--   1 guest   group1        696 Aug 23 09:51 .cshrc
Unix: -rw-r--r--   1 guest   group1        562 Nov 22  1999 .profile
Unix: -rw-r--r--   1 guest   group1        889 Jan 17 16:38 beta.pds
Unix: -rw-r--r--   1 guest   group1        841 Jan 17 16:31 earth.ua
Unix: -rw-r--r--   3 guest   group1        512 Jan 17 16:33 earth.unixua
Unix: -rw-rw-rw-   1 guest   group1        225 Jan 17 16:34 ocean.unixua
Unix: -rw-r--r--   2 guest   group1        512 Jan 17 16:35 world.unixua
zost>
```

File transfer is just as easy in the other direction. To move a file from the remote host to the local host, use the RECEIVE command. The example below transfers the file "LOGIN.COM" from the remote system to the local IBM z/OS host.

```
zost> receive login.com
eFT: Source                        Destination                    Size
eFT: ----------------------------  -----------------------------  ----
eFT: /home/guest/login.com         GUEST1.LOGIN.COM                967
zost>
```

This transfer can be verified by viewing a **LOCAL DIRECTORY** (**LISTCAT**) listing.

```
zost> local directory
IN CATALOG:USERCAT1
GUEST1.BATCH.DATA
GUEST1.BATCH.JCL
GUEST1.EARTH.UA
GUEST1.LOGIN.COM
GUEST1.MVS.PDS
GUEST1.OCEAN.UA
GUEST1.WORLD.UA
zost>
```

To force a disconnection from all remote hosts (in this case the UNIX host), the **EXIT** command is used. **EXIT** insures a smooth shut down of network activities as well as local and remote files. **EXIT** also returns a NetEx/eFT session status that can be interpreted by the local IBM z/OS host. This status is especially useful when NetEx/eFT is used within a JCL stream.

```
zost> exit
READY
```

To keep this sample session short, no more commands or features of NetEx/eFT will be shown. However, since only a small fraction of NetEx/eFT has been described here, the user is encouraged to read the remaining sections for a full description of the benefits that can be realized using NetEx/eFT.

# IBM z/OS Local User's Guide

## Introduction

This section is intended for users that would like an introduction to NetEx/eFT and some of its features. This section explains how to invoke NetEx/eFT from a z/OS TSO/E terminal, what a NetEx/eFT session looks like, logging in and transferring files to a remote host on the network, and executing commands on a remote host. Users are encouraged to refer to "Advanced Local User's Guide" on page 63 for a more in-depth look into NetEx/eFT. Users should also refer to the Software Reference Manual for the remote host to which a connection will be made for additional information about that host's environment.

## Invoking NetEx/eFT on IBM z/OS

NetEx/eFT is invoked using the following general format.

### Starting the NetEx/eFT Client (Initiator) Program

Log on to TSO/E according to your installation procedures. Issue the TSO/E command:

```
[%] | EFTUSER |  [SCRIPT([')  [[[ '']'']  input-file  [''['']]
```

Where:

**EFTUSER** is the name of the TSO/E CLIST used to invoke NetEx/eFT. It is possible that these commands may conflict with another TSO/E command or CLIST already set up at a particular site. If NetEx/eFT is not invoked by either of these commands, contact the site administrator. If your terminal emulator supports the displaying of "[" and "]", add `TRAN(' ')` to prevent the character translation when help commands are display. If this is set incorrectly some erroneous characters will be displayed on the help screens.

**input-file** is an optional NetEx/eFT input or script file containing NetEx/eFT commands that may be read and executed. When NetEx/eFT completes execution of the input file, the session terminates the TSO/E system prompt is displayed.

**argument1, argument2, ...** are optional arguments that may be passed as parameters to the input file. Multiword arguments should be enclosed in double quotation marks.

**-keyword value** (optional) specifies optional command line keywords that may be given to affect operation of the NetEx/eFT session. The following are valid keywords:

- **-ADAPTer** specifies a NetEx (H210) subsystem name for use by the **CONNECT** command (displayed as the **ADAPTer** keyword in **SHOW CONNECT**).
- **-BLOCKsize** specifies the size in bytes of the **CONNECT** block size (displayed as the **BLOCKsize** keyword in **SHOW CONNECT**). The default value is 16384.
- **-GLObal** specifies the size in bytes of the global variable environment (displayed as the environment variable "USER_GLOBAL"). The default value is 3000 bytes which should be adequate unless a user session attempts to define a large number

of global variables, in which case the **GLOBAL** switch can be used to increase the space available for global variables.

**-HOMEdir** specifies the name of the user's "login" or "home" directory when NetEx/eFT is invoked (displayed as the environment variable "USER_HOME"). Changing this keyword's value redefines the location NetEx/eFT uses to locate user startup files.

**-OUTput** specifies the name of an output file that is to receive the output from this session.

**-ROOTdir** specifies the name of the installed NetEx/eFT root directory containing the site-specific message, help, and startup files. This keyword is displayed as the environment variable "USER_ROOT". There is generally no reason to modify this keyword.

**-SEArch** specifies the search path NetEx/eFT follows to locate local initiator startup files. SEARCH is described in more detail in "Local IBM z/OS NetEx/eFT Startup Files".

**-SERvice** specifies an alternative default **CONNECT** service name. This keyword is displayed as the environment variable **USER_SERVICE**. The default is "EFT".

**-SSLCIPHERS** specifies which ciphers to use. It is a list of four-digit numbers. If omitted, all ciphers are available. Refer to the IBM document "*z/OS Cryptographic Services System SSL Programming*" for the four-digit codes

**-SSLPROTOCOL** specifies which protocols to use. "ALL" allows all protocols to be used. "ALL:-tlsv1" allows all protocols except TLSV1 may be used. The valid protocols are: ALL, TLSV1, TLSv1.1 TLSv1.2. Addition may also be use like: "TLSv1.1+TLSv1.2"

**-SECURE** specifies the connection will be encrypted. The Service must match this setting. If **SECURE** is not specified, the connection will not be encrypted.

## Examples

```
%EFT
%EFT SCRIPT(input.file)
%EFT SCRIPT('-global 4096')
%EFT SCRIPT('input.file -global 4096')
%EFT SCRIPT('input.file arg1 "arg 2"')
%EFT SCRIPT('input.file arg1 "arg 2" -global 4096')
%EFT SCRIPT(''userid input.file''')
%EFT SCRIPT(' ''''userid input.file'''' -global 4096')
%EFT SCRIPT(' ''''userid input.file'''' arg1 "arg 2" ')
%EFT SCRIPT(' ''''userid input.file'''' arg1 "arg 2" -global 4096')
%EFT SCRIPT(DD:INPUT)
```

Once NetEx/eFT is invoked for interactive use, output similar to the following should appear:

```
%EFTUSER - EFT starting (TCP)
eFT:
eFT: Welcome to MVS EFT version 5.5.0-nnn N24
eFT:
eFT>
```

The NetEx/eFT prompt in the example above is **eFT>**, although NetEx/eFT may be configured to prompt with a different string. The prompt means that NetEx/eFT is ready to accept commands.

Experienced users who want to start the client program with an initial input script may do so in any of the following four ways:

1. Before starting the client program, you can create a sequential data set with the name "prefix.CLIENT.UA" where prefix is the TSO/E user prefix. NetEx/eFT will look for this data set name during startup and automatically process the commands contained therein.
2. Pass the client program an unqualified data set name on the **%EFT** command using the **SCRIPT** keyword. To pass the data set named "prefix.USER.DATA" where prefix is the TSO/E user prefix, issue the TSO/E command:

   ```
   %EFT SCRIPT(USER.DATA)
   ```

3. Pass the client program a fully qualified data set name on the **%EFT** command using the **SCRIPT** keyword. To pass the data set named 'EFT.USER.DATA', issue the TSO/E command:

   ```
   %EFT SCRIPT('''EFT.USER.DATA''')
   ```

   Note that three apostrophes (single quotes) are needed on both ends of a fully qualified data set name.
4. Pass the client program a DD statement name on the **%EFT** command using the **SCRIPT** keyword. To pass an input script previously allocated to the DD statement named "SCRIPT", issue the TSO/E command:

   ```
   %EFT SCRIPT(DD:SCRIPT)
   ```

## Local IBM z/OS NetEx/eFT Startup Files

When NetEx/eFT is invoked, it attempts to read two startup files on the local host: a site startup file which is actually a partitioned data set member named "SCLIENT" located in the NetEx/eFT root directory "(SITE)" data set, and a user startup file (sequential data set) catalogued under in the user's TSO/E prefix called "prefix.CLIENT.UA". The site startup file is read first then the user startup file is read. Neither of the startup files is required.

The startup files consist of NetEx/eFT commands. Typically, a site administrator will create the site startup file to define basic aliases for general users. The user startup file provides more sophisticated users with a way to define custom aliases and qualifier defaults. User startup files make it possible to override defaults in the site startup file. For example, a simple startup file could contain the lines:

```
* My startup file (this is a comment line)
*
set alias ld local directory
set alias rd remote directory
set local prefix MYHOST:
```

This startup file creates two NetEx/eFT aliases for displaying the local and remote directory listings, **ld** and **rd** respectively. It also sets the default NetEx/eFT local prefix to be `MYHOST:`. After NetEx/eFT is invoked, these new definitions will be read in, whether they are in the site startup file or the user startup file. They are available to the user as soon as the NetEx/eFT input prompt appears.

It is possible to invoke NetEx/eFT by declaring alternative startup files. This is done using the **SEARCH** qualifier on the command line when NetEx/eFT is invoked. By default, **SEARCH** is defined as "(SITE) (USER)". By implication, this reads partitioned data set (PDS) member "SCLIENT" from the local NetEx/eFT "(SITE)" partitioned data set and then "prefix.CLIENT.UA", in that order. The order can be changed, other file names may be specified, or the special **SEARCH** keyword "(NONE)" can be used to override the default. Refer to "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106 for more information.

## Remote NetEx/eFT Startup Files

In addition to the local startup files, there are equivalent remote startup files that the NetEx/eFT Responder on the remote host returns to the local Initiator following a successful connection. By default, both a site and user startup files are processed, but this can be overridden by the **CONNECT** command's **SEARCH** qualifier. Following a network connection, these startup files, if they exist, are sent back to the Initiator to be processed. They are not executed on the remote host. (For security reasons, the following commands may not be executed from a remote server startup file: **CONNECT**, **DISCONNECT**, **LOCAL**, **RECEIVE**, **REMOTE**, and **SEND**.) Any aliases defined in these files become available to the local user. This is important in that a NetEx/eFT alias defined in the remote startup file will override an alias that has been previously defined in the session. Whether or not this is desirable depends upon the situation; care must be taken when defining aliases in a remote startup file.

For a z/OS host the server site startup file is a partitioned data set named "SSERVER" in the NetEx/eFT root directory data set, and a user startup file (sequential data set) catalogued under the user's TSO/E prefix called "prefix.SERVER.UA". The site startup file is returned first, then the user startup file. Neither is required.

The exact name and location of the remote startup files depends on the remote host in which a connection is being made. Refer to the manual for the remote host for more information.

# Getting Started

Once the prompt appears, it is time to begin giving commands to NetEx/eFT. This section will present some basic concepts that are an important foundation for understanding the details of NetEx/eFT.

## NetEx/eFT Commands and Command Qualifiers

A NetEx/eFT command can be invoked anytime the command line prompt appears. Commands may be fully spelled out or abbreviated. The minimum spelling of any command is the first 'n' capital letters of the command name. Abbreviations for each command are shown in "Command Descriptions" on page 117.

Several of the NetEx/eFT commands have qualifiers or keywords associated with them. A command's qualifiers can affect how a command responds to a user, the performance of a command, and the flexibility of a command. Most of the qualifiers have default values already associated with them. The novice user does not need to be concerned with overriding or redefining these values. The sophisticated user can use the qualifiers to modify commands, often making the commands more powerful. There are two methods for changing the values of qualifiers:

1. A qualifier can be redefined to assume a new default value by means of the **SET** command.
2. The current value of a qualifier can be overridden by specifying a new value on the command line. This is accomplished by using the special character dash '-' followed by the qualifier name and its new value.

Command qualifiers are similar to NetEx/eFT commands in that they may be abbreviated. The minimum spelling of any qualifier is the first 'n' capital letters of the qualifier name. Abbreviations for each command qualifier are shown in "Command Descriptions" on page 117. For instance, the minimum spelling of the qualifier "**CREate**" is "**CRE**".

### Displaying the Valid Qualifiers for a Command

A list of valid qualifiers for a NetEx/eFT command can be obtained with the **SHOW QUALIFIER** command. The list also includes a brief description of each qualifier. For example, to display the list of valid qualifiers for the **INPUT** command, type:

```
eFT> show qualifier input
eFT:
eFT:   CONTinue .... continue on error (on/off)
eFT:   ECHO ........ echo input to terminal (on/off)
eFT:   PROMPT2 ..... secondary prompt for input continuation
eFT:   PROMpt ...... prompt string for USER input
eFT:   SEArch ...... search path for default INPUT commands
eFT:   VERify ...... verify string/alias substitution (on/off)
eFT:
```

## Displaying the Current Value of a Qualifier

The **SHOW** command is used to obtain a listing of the current values for a command's qualifiers. For example, a listing of the **SEND** qualifier values is displayed by entering:

```
zost> show send
13:45:16 MVS:
13:45:16 MVS:  BLOCKsize ......... 3120
13:45:16 MVS:  CHEcksum .......... off
13:45:16 MVS:  COMPress .......... off
13:45:16 MVS:  CRC ............... off
13:45:16 MVS:  CREate ............ new
13:45:16 MVS:  DDname ............
13:45:16 MVS:  DDname ............
13:45:16 MVS:  DELete_on_error ... off
13:45:16 MVS:  DEVICE_Wait .......
13:45:16 MVS:  DEVICE_Wait .......
13:45:16 MVS:* DIRectory:LOCal ... TEST1
13:45:16 MVS:* DIRectory:REMote .. TEST1
13:45:16 MVS:  DISPosition ....... CATALOG
13:45:16 MVS:  EXPand ............ off
13:45:16 MVS:  EXPIRation ........
13:45:16 MVS:  FLOW .............. off
13:45:16 MVS:  KEEPBLanks ........ off
13:45:16 MVS:  LABELnumber .......
13:45:16 MVS:  LABELTYPE .........
13:45:16 MVS:  MAXRECord ......... 10000
13:45:16 MVS:  METHod ............ lzw
13:45:16 MVS:  MODe .............. character
13:45:16 MVS:  PARTialrecord ..... on
13:45:16 MVS:  PDSE .............. off
13:45:16 MVS:  QUIet ............. off
13:45:16 MVS:  RECALL ............ on
13:45:16 MVS:  RECFORmat ......... VB
13:45:16 MVS:  RECLENgth ......... 251
13:45:16 MVS:  REFER ............
13:45:16 MVS:  RELEase ........... on
13:45:16 MVS:  RETAIN ............ off
13:45:16 MVS:  RETENtion .........
13:45:16 MVS:  RNT ............... off
13:45:16 MVS:  RNT_BUFalloc ...... 262144
13:45:16 MVS:  RNT_INTerval ...... 60
13:45:16 MVS:  RNT_TIMeout ....... 1200
13:45:16 MVS:  SPACe ............. *16k
13:45:16 MVS:  STATistics ........ on
13:45:16 MVS:  TAB ............... 0
13:45:16 MVS:* TSOPREfix:LOCal ... TEST1
```

```
13:45:16 MVS:* TSOPREfix:REMote .. TEST1
13:45:16 MVS:  UNIT ..............
13:45:16 MVS:  VOLume ............
13:45:16 MVS:  WRAP .............. on
13:45:16 MVS:
13:45:16 MVS:* Informational qualifier (cannot be modified).
13:45:16 MVS:
zost>
```

The qualifier name appears in the left-hand column and its value appears in the right-hand column. In this example, the value of qualifier **CREATE** is currently set to "new". Qualifier **QUIET** is turned "off". Notice that some qualifiers are flagged as "informational qualifiers". These are shown along with the **SEND** qualifiers but are not controllable in the same way. They appear because they provide information important to the command and the one using it. Qualifiers flagged as informational cannot be modified. (**DIRectory:LOCal** and **DIRectory:REMote**, shown above, may be modified using **SET LOCAL DIRECTORY** and **SET REMOTE DIRECTORY** respectively. The **SEND** and **RECEIVE** commands list them as informational qualifiers since they are used to direct file lookup for file transfers.)

An individual qualifier's value can be examined by using the **SHOW** command followed by the command name and qualifier name. For instance, the current value of the **INPUT PROMPT** qualifier can be shown by entering:

```
zost> show input prompt
13:47:10 MVS:PROMpt ............ {dfn(host:remote, host:remote, "eFT")}{"> "}
zost>
```

## Setting a Command Qualifier

Use the **SET** command to redefine the value of a qualifier for a command for the duration of the NetEx/eFT session or until it is changed again using the **SET** command. For example, to change the default **RECEIVE** file transfer mode to **STREAM**, modify the **MODE** qualifier of the **RECEIVE** command:

```
eFT> set receive mode stream
```

The **RECEIVE** file transfer mode now will default to **STREAM** until the qualifier **MODE** is redefined. The change can be verified with the command:

```
eFT> show receive mode
eFT: MODe .............. stream
```

Some command qualifiers, such as **INPUT** qualifiers **CONTINUE**, **ECHO**, and **VERIFY**, are Boolean qualifiers: their values are either **ON** or **OFF**. To set a Boolean command qualifier to **ON**, enter:

```
eFT> set command qualifier on
```

or

```
eFT> set command qualifier
```

For the **INPUT** qualifier **ECHO**, this would be:

```
eFT> set input echo on
```

or

```
eFT> set input echo
```

For Boolean qualifiers, a missing value is interpreted by NetEx/eFT as **ON**.

Besides string and Boolean qualifiers, there are also integer qualifiers. These qualifiers, such as **BLOCKSIZE**, **LINES**, and **TIMEOUT**, accept only integer values and often have numeric range checks associated with them.

Integer qualifier values may be appended with a 'K' ($2^{10}$) or 'M' ($2^{20}$) multiplier. For example, to set the **CONNECT BLOCKSIZE** qualifier to 16 kilobytes, the following may be entered:

```
eFT> set connect blocksize 16k
```

### Overriding a Command Qualifier

The qualifiers that can be defined with the **SET** command (all non-informational qualifiers), can also be overridden on the command line. For example, if the current **RECEIVE** file transfer mode is **STREAM**, it can be overridden for a single transfer by entering:

```
eFT> receive -mode character sourcefile
```

This command does not change the default value of the **MODE** qualifier, it simply overrides the default value for the duration of the command. Therefore, the file 'sourcefile' above would be transferred in **CHARACTER** mode while the default value of **RECEIVE** qualifier **MODE** would remain **STREAM**. This can be verified with the command:

```
eFT> show receive mode
eFT: MODe ............. stream
```

When forcing a Boolean qualifier to **ON** from the command line, the value **ON** is optional. For example, the commands shown below are equivalent.

```
eFT> send -quiet on sourcefile
eFT> send -quiet sourcefile
```

NetEx/eFT interprets the missing Boolean value to be **ON**, even if the default value is **OFF**.

## Online Help

Built into NetEx/eFT is an online help facility that makes it easy for a user to obtain help on a command or topic. The help facility also returns useful information on command qualifiers, qualifier defaults, and command examples. To obtain a general NetEx/eFT help display, use the **HELP** command as follows:

```
eFT> help
```

The general, or top-level help display will include additional topics in which help can be obtained. For instance, one of the help sub-topics will be the NetEx/eFT command **LOCAL**. To get additional help on the **LOCAL** command, one would type:

```
eFT> help local
```

To get help on qualifiers for the **LOCAL** command, one would type:

```
eFT> help local qualifiers
```

It is important to note that some help information resides on remote hosts. Therefore, a remote connection is required in some cases (such as **HELP SEND QUALIFIERS**).

Refer to the **HELP** command in "Command Descriptions" on page 117 for more details.

## Controlling NetEx/eFT Input and Output

The NetEx/eFT commands **INPUT** and **OUTPUT**, along with their respective qualifiers, control a majority of the user-oriented input and output within NetEx/eFT. By setting various qualifiers, users can change the NetEx/eFT prompt, tell NetEx/eFT to continue processing even if an error occurs, cause output to be held after each page, save the output to a local file, etc. This section very briefly discusses some of the things that can be done to control NetEx/eFT I/O.

By typing **SHOW INPUT**, the user can get a list of all **INPUT** qualifiers along with their current values:

```
eFT> show input
eFT:
eFT:   CONTinue ......... off
eFT:   ECHO ............. off
eFT:   PROMPT2 .......... More>>
eFT:   PROMpt ........... eFT>
eFT:   SEArch ...........
eFT:   VERify ........... off
eFT:
```

Each of these qualifiers is explained in detail in “Command Descriptions” on page 117 under the **INPUT** command, along with examples of its use. Very simply, the **SET** command is used to modify any of the qualifiers. For instance, to change the NetEx/eFT prompt from “eFT” to “MY-PROMPT:” type the following:

```
eFT> set input prompt "MY-PROMPT:  "
MY-PROMPT:
```

Notice that the prompt for the next command has now changed to “MY-PROMPT:”. To tell NetEx/eFT to continue processing within an input script or alias (discussed later) even after an error results, turn on the **CONTINUE** qualifier by entering:

```
eFT> set input continue on
```

Users can affect the output as it is returned from NetEx/eFT by modifying **OUTPUT** qualifiers. To look at the available qualifiers for the **OUTPUT** command, type **SHOW OUTPUT** command:

```
eFT> show output
eFT:
eFT:   COLumns ........... 80
eFT: * DESTination .......
eFT:   FORmat ............ {msg("text")} ({msg("facility")}{msg("code")}).
eFT:   HOLD .............. off
eFT:   INTernal .......... off
eFT:   LINes ............. 24
eFT:   PREFix ............ eFT:
eFT:   QUIet ............. off
eFT:   TRUNcate .......... off
eFT:
eFT: * Informational qualifier (cannot be modified).
eFT:
```

Each of these qualifiers is explained in detail in “Command Descriptions” on page 117 under the **OUTPUT** command, along with examples of their use. As with the **INPUT** qualifiers, the **SET** command can be used to modify any of the **OUTPUT** qualifiers. For example, to tell NetEx/eFT to pause every 24 lines (the current value of the **LINES** qualifier), turn on the **HOLD** qualifier with the following command:

```
eFT> set output hold on
```

This will prevent general NetEx/eFT output from scrolling off the screen. To modify the number of lines per screen to twenty, change the **LINES** qualifier:

```
eFT> set output lines 20
```

The **OUTPUT** command itself can be used to capture the results of a NetEx/eFT session to a file. This is done by typing **OUTPUT** followed by a file name. In addition, the user’s input can be captured by turning on the **INPUT ECHO** qualifier:

```
eFT> set input echo on
eFT> output tmpfile
```

Following this command sequence, all input and output for this session is directed to the file named "tmpfile". If the **ECHO** qualifier was not turned on, only the command results (output) would be captured. More information concerning **INPUT** and **OUTPUT** can be found in the "Advanced Local User's Guide" on page 63 of this manual. This facility is particularly useful as a means of providing information to Network Executive Software's technical support personnel regarding questions and problems.

## NetEx/eFT Error Messages

NetEx/eFT provides a friendly user interface across many different host types. This includes error messages that are easy to understand. Error messages returned by NetEx/eFT consist of at least a NetEx/eFT level error message followed by an optional host specific error message. All error messages also have an associated error code that can be used to locate additional information in the error message appendices.

An example of a simple "Invalid command" error follows:

```
eFT> xxxxxx
eFT: Invalid command 'xxxxxx' (UA-4708).
```

The error text is straightforward. The error code "(UA-4708)" indicates the error is a general NetEx/eFT error with error number 4708.

The next example demonstrates an error resulting from a **SEND** command that contains a general NetEx/eFT error followed by a host specific NetEx/eFT error and finally an operating system specific error:

```
eFT> send badfile
eFT: Failure during CHARACTER mode send (UA-5001).
eFT: Failed to access file 'badfile' (UA123-8302).
eFT: OS - file not found (OS-18012).
```

The first error code "(UA-5001)" indicates that this is a general NetEx/eFT error (UA) with an error number of 5001. The second error code "(UA123-8302)" says the error is from NetEx/eFT (UA), but generated by the NetEx/eFT product number 123 (or more exactly H123). The actual error number is 8302. The last error code "(OS-18012)" indicates the error is generated by the operating system (OS or whatever the operating system name might be), with the operating system error number of 18012. The NetEx/eFT error messages are listed in "Appendix B. NetEx/eFT Error Messages for IBM z/OS" on page 216 and in similar appendices in other NetEx/eFT manuals. The general NetEx/eFT errors can be found in any manual. The product specific errors are in the manual for the product indicated by the product number (e.g., UA123 is product H123). Refer to the appropriate manuals for the host operating system for any operating system messages.

It is important to note that a site has the ability to change the error message format and it may not exactly match the examples above. There are, however, three main pieces of information for each message: the message text, the facility generating the message, and the error number. This information should be easy to decipher. If not, see the site administrator.

## Aliasing

Much of the versatility NetEx/eFT offers for users is based on a very powerful script-processing or alias capability. Users of the product benefit from aliasing by having special commands, or aliases, defined for them. While a detailed description of the facility is provided in "Advanced Local User's Guide" on page 63, this brief discussion is provided to give a general familiarity of aliasing without getting lost in detail.

Aliasing provides a means of creating a custom command set for a user or group of users. An alias is nothing more than a new name for a NetEx/eFT command or set of commands. Aliases are useful for creating "shorthand" commands for complex or frequently used NetEx/eFT command sequences. The simplest aliases are one for one translations of an alias name and a NetEx/eFT command. For example, if the user is accustomed to

typing a question mark to obtain help in an application, an alias can be defined very easily using the **SET ALIAS** command to map "?" to **HELP**. The new alias may then be viewed with the **SHOW ALIAS** command.

```
eFT> set alias ? help
eFT> show alias ?
eFT: ? ........... HELP
```

Now, instead of typing **HELP** to obtain help information, the user can just type "?" at the NetEx/eFT prompt. The commands are considered equivalent by NetEx/eFT. Below is the definition of a much more complicated alias called **EDIT** which allows a user to use a familiar local editor to edit a remote file.

```
eFT> set alias EDit {} -
eFT>           receive -mode character {1} edit.tmp !
eFT>           local -interactive myeditor edit.tmp !
eFT>           send -mode character -create replace edit.tmp {1} !
eFT>           local delete edit
```

The basic procedure of the **EDIT** alias is to transfer the remote file to the local host (**RECEIVE**), edit the temporary file using the local editor (**LOCAL -INTERACTIVE MYEDITOR**), send the file back to the remote host when the edit is complete (**SEND**), and finally delete the temporary file (**LOCAL DELETE**). The exact syntax and special characters used to define the alias are explained in detail in "Developing NetEx/eFT Scripts Using Input Files and Aliases" on page 92.

To use the alias, the user simply invokes it from the command line like any other NetEx/eFT command. For example, to edit an existing file on the remote host called "MYFILE", you type:

```
eFT> edit myfile
```

NetEx/eFT takes care of the rest. Even though several NetEx/eFT commands are required to edit a remote file, the user sees it as a simple **EDIT** command. This is the real advantage of aliasing.

To display the definition of the **EDIT** alias, the **SHOW ALIAS** command is used:

```
eFT> show alias edit
eFT: Edit ........ receive -mode character {1} edit.tmp
eFT:               local -interactive myeditor edit.tmp
eFT:               send -mode character -create replace edit.tmp {1}
eFT:               local delete edit.tmp
```

Aliases created within an interactive session are lost when the session is terminated. To create aliases that can be used from session to session, they must be defined within a NetEx/eFT input or script file, or within a site or user startup file which are read automatically when NetEx/eFT is invoked. Refer to "Developing NetEx/eFT Scripts Using Input Files and Aliases" on page 92 for a detailed description of aliasing.

## Terminating a NetEx/eFT Session

To end an interactive NetEx/eFT session type **EXIT**:

```
eFT> exit
```

**EXIT** will disconnect all connections to remote hosts and terminate the current NetEx/eFT session. Any local or remote files that had been opened will be closed. The **QUIT** command also may be used to terminate an interactive session. Refer to "Command Descriptions" on page 117 for more details on **EXIT** and **QUIT**.

# Establishing a Connection to a Remote Host

In order to transfer files or execute commands on another host, a network connection must be established. This connection provides a link between the NetEx/eFT Initiator on the local host and the NetEx/eFT Responder on the remote host. There are three ways to make a connection to a remote host; the **CONNECT** command or the **LOGIN** and **LOGON** aliases.

## Using CONNECT to Establish a Connection

The **CONNECT** command allows a user to login to a remote host. The basic format of the command is:

| Command | Parameters |
|---|---|
| CONnect | host userid password [parameters] |

Where:

**host** is the name of a remote host as defined in the local network.

**userid** is the username or ID describing a valid user account on that host.

**password** is the associated password needed to login to userid.

**parameters** indicates additional parameters that may be required by the remote host at login time.

Below is an example **CONNECT** where the host name is "Yellowstone", the userid is "guest", and the password is "Netex". The connection is to be encrypted:

```
eFT> connect yellowstone guest netex -serv efts -secure
13:51:29 MVS:Connected to Service Initiator on host 'yellowstone'.
========================================================================
Last login: Thu Jun  6 13:27:01 CDT 2019 from zpdt2-5.netexsw.com on pts/5
Directory: /mnt/home2/test1
Thu Jun  6 13:51:58 CDT 2019
========================================================================
13:52:01 MVS:Logged in as user 'test1'.
13:52:01 MVS:Connected to service '54078' on host 'yellowstone'.
13:52:01 MVS:Welcome to UNIX NETEX-eFT version 5.5
```

Following a successful **CONNECT**, NetEx/eFT returns several informative messages, the exact syntax of which depends upon the host to which a connection is being made. The first message above indicates that an initial network connection was established to the NetEx/eFT Responder (Service Initiator or service "EFT"). Following that message are several lines of information surrounded by equal signs (===). The information between the equal signs is returned by the remote operating system at login time. This information is not necessarily important to NetEx/eFT but may be to the user logging in. Next is a NetEx/eFT message indicating that a successful login has completed. Finally, a message may appear that informs the user of the name of the network service handling the connection.

Besides the additional parameters that can be passed directly to the remote login procedure, the **CONNECT** command also has several qualifiers associated with it. The use of most of these qualifiers is a function of the remote host. Refer to the User's Guide for the remote host for more information. "Command Descriptions" on page 117 describing the **CONNECT** command will also assist in the use of this command and its qualifiers.

Since most users would rather be prompted for input and would rather not see their passwords echoed back to the terminal (if possible), it is suggested that the **LOGIN** and **LOGON** aliases be used when establishing a remote host connection. This alias is documented in the next section.

## Using LOGIN to Establish a Secure Connection

The suggested way to establish a remote connection is to use the **LOGIN** or **LOGON** alias. **LOGIN** and **LOGON** are similar to **CONNECT** but have the advantage of being interactive. Below is a repeat of the example from the previous section but using **LOGIN** instead of **CONNECT**:

```
eFT> login
Hostname? yellowstone
Username? guest
Password?
Qualifiers? -serv EFTS -SECURE
09:40:10 MVS:Connected to Service Initiator on host 'yellowstone'.
========================================================================
Last login: Thu Jun  6 09:40:12 CDT 2019 from zpdt2-5.netexsw.com on pts/5
Directory: /mnt/home2/test1
Thu Jun  6 09:47:28 CDT 2019
========================================================================
09:47:31 MVS:Logged in as user 'guest'.
09:47:31 MVS:Connected to service '51067' on host 'yellowstone'.
09:47:31 MVS:Welcome to UNIX NETEX-eFT version 5.5
yellowstone>
```

Notice that **LOGIN** prompts the user for appropriate login information and that the password was not printed to the terminal. (Whenever possible NetEx/eFT supports "NO-ECHO" mode to improve security; not all systems provide this mode.) This interface is much more friendly than using **CONNECT** and can be tailored to the needs of a given site by the system administrator. Following the prompts, the connect proceeds as expected.

**Note:** Since **LOGIN** is an alias that can be modified by the site administrator, it may operate differently than the example. However, the overall process should remain similar.

## Exchanging Host Information on Connect

To the user, the connect/login process appears straightforward, but to NetEx/eFT, much must be done in order for two hosts to communicate. The issues concerning **CONNECT** (**LOGIN**) qualifiers and login are addressed in the "Remote User's Guide" section of the manual for the host to which the connection is being made. Contained in this section is a general discussion on the information passed by NetEx/eFT that is available to the user. This information may be useful in making decisions once a connection has been established.

Once a successful login has been assured, the NetEx/eFT Responder (the remote server) sends information about itself to the Initiator (the local client) and vice versa. The information, which describes both the remote and local environments, is exchanged between the two sides to establish how compatible they are and what functions can be supported. The **SHOW** command is used to display this information. For instance, to display information describing the local environment, type **SHOW LOCAL** as:

```
yellowstone> show local
09:56:43 MVS:
09:56:43 MVS:* COPYRight ......... COPYRIGHT (c) 1999-2021 - Network Executive
Software, Inc.  Mpls. MN
09:56:43 MVS:  DIRectory ......... TEST1
09:56:43 MVS:* GATEway ...........
09:56:43 MVS:  HOMEdir ........... TEST1
09:56:43 MVS:* HOSTCODE .......... EBCDIC
09:56:43 MVS:* HOSTENV ........... TSO FOREGROUND
09:56:43 MVS:* HOSTOS ............ z/OS 2.3
09:56:43 MVS:* HOSTTYPE .......... MVS
09:56:43 MVS:* LicExp ............ 20200101
```

```
09:56:43 MVS:* LicKey ............ DSEY-YAAA-AAAY-5QUZ-4EGA-5KN7
09:56:43 MVS:* LicNotOper ........ 20200101
09:56:43 MVS:* LicProto .......... IP SSL
09:56:43 MVS:* NETwork ........... TCPIPS
09:56:43 MVS:* PID ............... 0XF9BE808BB360
09:56:43 MVS:* PRODuct ........... EFT213
09:56:43 MVS:* ROOTdir ........... EFT.EFT4.TCP.TEXT
09:56:43 MVS:* STATus ............ 0
09:56:43 MVS:  TSOPREfix ......... TEST1
09:56:43 MVS:* USERname .......... DAN3
09:56:43 MVS:* VERsion ........... 5.5.0-nnnn N24
09:56:43 MVS:
09:56:43 MVS:* Informational qualifier (cannot be modified).
09:56:43 MVS:
```

The qualifiers that are preceded by an asterisk (**HOSTCODE**, **PID**, etc.) reflect environmental data describing the local host and cannot be changed by the user. The remaining qualifiers (**DIRectory**, **PREFix**, etc.) that appear are directly tied to the **LOCAL** command and may be modified to affect that command's execution. (Note that the display above is only a sample of the information that might actually be seen for a particular host).

To display the remote environment's information, use the **SHOW REMOTE** command:

```
yellowstone> show remote
09:59:28 MVS:
09:59:28 MVS:* BLOCKsize ......... 16384
09:59:28 MVS:* COPYRight ......... Copyright (C) 1999-2021, Network Executive
Software, Inc.
09:59:28 MVS:  DIRectory ......... /mnt/home2/test1
09:59:28 MVS:* GATEway ...........
09:59:28 MVS:  HOMEdir ........... /mnt/home2/test1
09:59:28 MVS:* HOST .............. yellowstone
09:59:28 MVS:* HOSTCODE .......... ASCII7
09:59:28 MVS:* HOSTTYPE .......... UNIX
09:59:28 MVS:* LicExp ............ 20191130
09:59:28 MVS:* LicKey ............ DNLJ-YAC2-ABAA-CWOV-SOHV-5JH2
09:59:28 MVS:* LicNotOper ........ 20200228
09:59:28 MVS:* LicProto .......... IP SSL
09:59:28 MVS:* NODE ..............
09:59:28 MVS:* PID ............... 28860
09:59:28 MVS:  PREFix ............ Unix:
09:59:28 MVS:* PRODuct ........... EFT803
09:59:28 MVS:  QUIet ............. off
09:59:28 MVS:* ROOTdir ........... /usr/share/nesi/eftip/user
09:59:28 MVS:  SECure  ........... on
09:59:28 MVS:* SERvice ........... 51067
09:59:28 MVS:  SHELL ............. /bin/bash
09:59:28 MVS:* SSLCipher ......... AES256-SHA
09:59:28 MVS:* SSLFIPSmode ....... on
09:59:28 MVS:* SSLProto .......... TLSv1
09:59:28 MVS:* STATus ............
09:59:28 MVS:* TRANSlate ......... Network
09:59:28 MVS:* USERname .......... test1
09:59:28 MVS:* VERsion ........... 5.5
09:59:28 MVS:
09:59:28 MVS:* Informational qualifier (cannot be modified).
09:59:28 MVS:
```

Again, the qualifiers marked by an asterisk describe the remote environment (**HOST**, **PID**, etc.) as well as information important to the connection itself (**BLOCKsize**, **TRANSlate**, etc.). The remaining qualifiers (**DIRectory**, **QUIet**, etc.) are directly associated with the **REMOTE** command and affect its execution.

## Establishing Multiple Host Connections

A NetEx/eFT session may have up to ten host connections at any given time. Although ten may be unrealistic in most applications, it may be desirable from time to time to make a second host connection at the same time another connection is in place. For example, assume the user of the session below has already established a connection from the local host to a remote host named "YELLOWSTONE". This first connection can be verified by invoking the **SHOW HOST** command:

```
yellowstone> show host
10:11:43 MVS:
10:11:43 MVS:active --> (1) Host=yellowstone  User=test1
10:11:43 MVS
```

**SHOW HOST** gives a list of all existing connections for the present session. The current "active" connection is flagged. The active connection is the one, if any, that reflects the current remote host. To establish a second connection the **LOGIN** alias is used as explained in a previous section. For example, to connect to a host named "DOWNEY", the following command sequence is used:

```
eFT> login
Hostname? downey
Username? test2
Password?
Qualifiers? -serv 6900
10:15:00 MVS:Connected to Service Initiator on host 'downey'.
================================================================================
Last login: Thu Feb 22 13:12:27 from zpdt2-5

================================================================================
10:15:06 MVS:Logged in as user 'test2'.
10:15:06 MVS:Connected to service '43437' on host 'downey'.
downey>
```

The **SHOW HOST** command can be used again to display the list of connections held by this session:

```
downey> show host
10:15:55 MVS:
10:15:55 MVS:           (1) Host=yellowstone  User=test1
10:15:55 MVS:active --> (2) Host=downey    User=test2
10:15:55 MVS:
downey>
```

Notice that "DOWNEY" is now flagged as the active host. This means that any file transfer or remote command execution will be directed to it instead of host "YELLOWSTONE". The **SHOW REMOTE** command also will display the remote environment for host "DOWNEY" since it is now active. The connection to host "YELLOWSTONE" remains but is in an idle state. To make it the active connection, the **SET HOST** command is used as:

```
downey> set host yellowstone
yellowstone>
```

or:

```
downey> set host 1
```

Now a look at the host display will show that "YELLOWSTONE" is the active host:

```
yellowstone> show host
0:17:20 MVS:
0:17:20 MVS:active --> (1) Host=yellowstone  User=test1
0:17:20 MVS:           (2) Host=downey    User=test2
0:17:20 MVS:
yellowstone>
```

Having multiple host connections can be useful for managing system activities on several hosts from a single point. For instance, a user on one host can send messages to several other hosts. Or a user can start up jobs on several other hosts all from a single terminal on the network.

## Disconnecting from a Host

To terminate an existing connection, the **DISCONNECT** command is used. Assume two connections are currently established to hosts "YELLOWSTONE" and "DOWNEY" respectively, where "YELLOWSTONE" is the active connection. The following will terminate this connection:

```
yellowstone> disconnect
10:18:06 MVS:Disconnected from host yellowstone.
eFT>
```

To verify the connection has been broken, use the **SHOW HOST** command:

```
eFT> show host
10:18:12 MVS:
10:18:12 MVS:           (2) Host=downey    User=test2
10:18:12 MVS:
eFT>
```

Following a disconnect, there is no active host. In order to make an existing idle connection active, use the **SET HOST** command. The following command will make the connection to "DOWNEY" active:

```
eFT> set host downey
```

**SHOW HOST** will now indicate the change:

```
downey> show host
10:19:14 MVS:
10:19:14 MVS:active --> (2) Host=downey    User=test2
10:19:14 MVS:
downey>
```

An alternative way to disconnect from an active host is to exit the NetEx/eFT session. The **EXIT** command causes all connections to be disconnected prior to terminating the session.

# Transferring Files as a Local User

The file transfer capabilities of NetEx/eFT are provided by two commands, **SEND** and **RECEIVE**. The **SEND** command provides file transfer from a user's local host to the current remote host. The **RECEIVE** command transfers files from the remote host back to the local host. Prior to transferring files, a network connection must exist.

## Sending Files to a Remote Host

The basic format of the **SEND** command is:

| Command | Parameters |
|---|---|
| SEND | src_spec [dest_spec] [qualifiers] |

Where:

**src_spec** is the file specification of the local file to be transferred to the remote host.

**dest_spec** is the file specification of the remote file which is to be created or replaced by the transfer. This parameter is optional. If it is omitted, NetEx/eFT will use **src_spec** to create the destination file specification based on the remote host.

**qualifiers** represents optional **SEND** qualifiers that may be added to the command line to override the default values. The **SEND** qualifiers control such things as file creation, mode of transfer, and record orientation, and are defined by the remote host.

Once a connection to a remote host has been established, the user may begin transferring files. This is generally as easy as typing **SEND** followed by a local file name:

```
eFT> send src_spec
```

where **src_spec** is the name of an existing file on the local host. NetEx/eFT takes care of mapping the local file name to a valid remote file specification in all but a few instances. If NetEx/eFT cannot successfully handle the mapping, for example if the source file name contains unusual characters that the remote host just cannot tolerate, then the user must include the destination file name on the command line. Specifying the destination name is also useful for changing the name of a file from one host to another. The example below transfers file "src_spec" and renames it "new_file" on the remote host:

```
eFT> send src_spec new_file
```

The **SEND** command also supports wildcarding on both the source and destination file specifications. This information along with the host-specific information concerning file transfers, including examples, is explained in the file handling section of the appropriate manual. Source file specifications, source wildcarding, etc., can be found in "File Handling Under IBM z/OS NetEx/eFT" on page 41. Destination file specifications, destination wildcarding, and qualifiers that affect the **SEND** command, can be found in the same section of the manual for the host to which files are being transferred.

## Receiving Files from a Remote Host

The basic format of the RECEIVE command is:

| Command | Parameters |
|---|---|
| RECeive | src_spec [dest_spec] [qualifiers] |

Where:

**src_spec** is the file specification of the remote file to be transferred to the local host.

**dest_spec** is the optional specification of the local file which is to be created or replaced by the transfer. If it is omitted, NetEx/eFT will use **src_spec** to create the destination file specification on the local host.

**qualifiers** optional **RECEIVE** qualifiers that may be added to the command line to override the default values. The **RECEIVE** qualifiers are defined by the local host. As do the **SEND** qualifiers, the **RECEIVE** qualifiers control such things as file creation, mode of transfer, and record orientation.

Files can be received from a remote host as soon as a connection has been established. Receiving a file is as easy as typing **RECEIVE** followed by a remote file name:

```
eFT> receive src_spec
```

where "src_spec" is the name of a file that currently resides on the remote host. In the same way as it handles **SEND**, NetEx/eFT maps the remote file name to a valid local file name in all but a few instances which are generally due to character or length conflicts. If the file name mapping cannot be automated, or if the user simply wishes to rename the file as it is received, the local file name must be included as a second parameter on the command line, as shown:

```
eFT> receive remote_file local_file
```

The example above transfers file "remote_file" from the remote host and renames it "local_file" on the local host.

The **RECEIVE** command supports wildcarding on both the source and destination file specifications. This information along with the host-specific information concerning file transfers, is explained in the file handling section of the appropriate manual. Source file specifications, source wildcarding, etc., can be found in file handling in the manual for the remote host. Destination file specifications, destination wildcarding, and qualifiers that affect the **RECEIVE** command can be found in "File Handling Under IBM z/OS NetEx/eFT" on page 41 of this manual.

## Send and Receive Qualifiers

NetEx/eFT was designed to make file transfer very easy for all types of users. Much of the simplicity comes from the use of default qualifier values. Although **SEND** and **RECEIVE** have several qualifiers associated with them, defaults can be set up to operate most of the time for most users. Therefore, most users seldom need to modify the qualifier values. On the other hand, changing the value of a **SEND** or **RECEIVE** qualifier is simple.

Only simple validation of many of these qualifiers is performed when the set command is issued. Detailed validation is performed when the send or receive is issued.

To show the available **SEND** or **RECEIVE** qualifiers after establishing a remote connection, use the **SHOW QUALIFIERS** command. For example, to display the list of valid qualifiers for **SEND**, type the following:

```
eFT> show qualifiers send
eFT:
eFT:   CRC ............. file transfer checksum (on/off)
eFT:   CREate .......... file create options
eFT:   MAXRECord ....... maximum RECORD mode size
eFT:   MODe ............ file transfer mode
eFT:   QUIet ........... inhibit file transfer display (on/off)
eFT:
```

The output above reflects a sample of the many qualifiers that might be seen. The actual qualifiers for **SEND** depend on the remote host since that is where file creation takes place. The **RECEIVE** qualifiers are directly associated with the local host for the same reason. If a new connection is made to a different host, the qualifiers may change significantly.

To view the current values for the **SEND** or **RECEIVE** qualifiers, use the **SHOW** command. For example, **SHOW SEND** displays the list of **SEND** qualifiers along with their current values:

```
zos5> show send
eFT:
eFT:   BLOCKsize ......... 3120
eFT:   CHEcksum .......... off
eFT:   COMPress .......... off
eFT:   CRC ............... off
eFT:   CREate ............ new
eFT:   DDname ............
eFT:   DDname ............
eFT:   DELete_on_error ... off
eFT:   DEVICE_Wait .......
eFT:   DEVICE_Wait .......
eFT: * DIRectory:LOCal ... TEST
eFT: * DIRectory:REMote .. TEST1
eFT:   DISPosition ....... CATALOG
eFT:   EXPand ............ off
eFT:   EXPIRation ........
eFT:   FLOW .............. off
eFT:   KEEPBLanks ........ off
eFT:   LABELnumber .......
eFT:   LABELTYPE .........
eFT:   MAXRECord ......... 10000
eFT:   METHod ............ lzw
eFT:   MODe .............. character
eFT:   PARTialrecord ..... on
eFT:   PDSE .............. off
eFT:   QUIet ............. off
eFT:   RECALL ............ on
eFT:   RECFORmat ......... VB
eFT:   RECLENgth ......... 251
eFT:   REFER ............
eFT:   RELEase ........... on
eFT:   RETAIN ............ off
eFT:   RETENtion .........
eFT:   RNT ............... off
eFT:   RNT_BUFalloc ...... 262144
eFT:   RNT_INTerval ...... 60
eFT:   RNT_TIMeout ....... 1200
eFT:   SPAce ............. *16k
eFT:   STATistics ........ on
eFT:   TAB ............... 0
eFT: * TSOPREfix:LOCal ... TEST
eFT: * TSOPREfix:REMote .. TEST1
eFT:   UNIT ..............
eFT:   VOLume ............
eFT:   WRAP .............. on
eFT:
eFT: * Informational qualifier (cannot be modified).
eFT:
zos5>
```

Notice that a **DIRECTORY** entry appears for both the local and remote host. This value determines where the file will come from and where it will be sent if the respective file specifications are not given. (These qualifiers may be modified by using **SET LOCAL DIRECTORY** and **SET REMOTE DIRECTORY**.) The remaining qualifiers (the non-informational qualifiers) may be modified using the **SET** command. For example, to change the **RECEIVE** command's default file option **CREATE** from **NEW** to **REPLACE**, use the following:

```
eFT> set receive create replace
```

Or, to override the current value for a single file transfer, modify it on the **SEND** or **RECEIVE** command line. For example:

```
eFT> receive sourcefile -create replace
```

For a complete list of valid **RECEIVE** qualifiers, refer to "File Handling Under IBM z/OS NetEx/eFT" on page 41. This section will also include detailed information about transferring files to this host, wildcard support, transfer modes, and much more. Refer also to the **RECEIVE** command in "Command Descriptions" on page 117 of this manual.

The qualifiers for the **SEND** command on the other hand, are detailed in the file handling and command description sections of the manual for the remote host to which file transfers will be made. That manual will also address information concerning host file specifications, wildcard support, file types supported, etc.

# Executing Remote Host Commands

NetEx/eFT users can issue host commands on the remote host and view the results. Host commands can take the form of a native host command or an alias that translates to a host-specific command. Remote commands are issued from a NetEx/eFT session via the **REMOTE** command. A network connection to a remote host must exist prior to issuing **REMOTE**. The command line format is:

| Command | Parameters |
|---|---|
| REMOTE | [qualifiers] command |

Where:

**REMOTE** is the keyword for this command.

**qualifiers** optional NetEx/eFT **REMOTE** qualifiers that may be added to the command line to override the default values. These qualifiers and their default values are defined by the remote host. Qualifiers must appear before the remote command.

**command** a valid command on the remote host, an alias command defined using **SET REMOTE ALIAS**, or one of the predefined host independent commands.

NetEx/eFT performs translation on any alias prior to passing the command string to the remote host. By default, the results of a **REMOTE** command get transferred back across the network and displayed at the local user's terminal.

For example, assume the remote host supports a command called "DISPLAY TIME" that returns the current time of day. A user could execute this command from a NetEx/eFT session by typing the following:

```
eFT> remote display time
SYSTEM-A:
SYSTEM-A:  The current time is: 12:12:01 pm
SYSTEM-A:
```

The results are displayed in the remote host's format prepended with the optional host prefix that precedes each line of output ("SYSTEM-A:"). This prefix can be modified to the user's liking with the **SET REMOTE PREFIX** command.

Since users may be unfamiliar with the command syntax of a remote host, NetEx/eFT defines a set of commands (implemented as aliases) that exist on all hosts[1]. These commands, referred to as host-independent commands, allow a user to execute commands on many different systems with a single, simple command set. To look at the list of host independent commands defined for the current remote host, issue the **SHOW REMOTE ALIAS** command:

```
eFT> show remote alias
eFT:
eFT: COPY ............. Copy_a_file
eFT: DELete ........... Delete_a_file
eFT: DIRectory ........ List_files
eFT: REName ........... Rename_a_file
eFT: TYPe ............. Type_contents_of_a_file
eFT: WHO .............. Who_is_on_the_system
```

The actual output seen by the user will list all of the remote aliases (including host independent commands) in the left column and the host command translations in the right column. Users can issue host-independent commands as if they were commands native to the remote host. NetEx/eFT handles the translation. For example, to obtain a list of files that reside on the remote host, the host-independent command **DIRectory** could be used:

```
eFT> remote dir
```

The actual native command for the remote host could be given. Assuming the native command for listing files on the remote host is "LISTFILES", an alternative to the above would be:

```
eFT> remote listfiles
```

The commands would give identical results since the host independent command **DIRectory** would be mapped to the native command "LISTFILES" for this host.

Refer to the remote user's guide in the manual of the remote host for a list of host-independent commands defined for that system, as well as a discussion on executing commands on that host. Also see the command description section of the same manual for the list of valid **REMOTE** qualifiers and an example of their use.

# Executing Local IBM z/OS Commands

Users can issue host commands on the local host and view the results. Host commands can take the form of a valid TSO/E command or an alias that translates to a valid TSO command. Local commands are issued from a NetEx/eFT session via the **LOCAL** command. The format of the command is:

| **Command** | **Parameters** |
|---|---|
| LOCAL | [qualifiers] [command] |

Where:

**qualifiers** optional NetEx/eFT **LOCAL** qualifiers that may be added to the command line to override the default values. Qualifiers must appear before the local command.

**command** a valid TSO/E command or CLIST, an alias command defined using **SET LOCAL ALIAS**, or one of the predefined host independent commands (e.g. **DIRECTORY**, **TYPE**, **WHO**, etc.). NetEx/eFT performs translation on any alias prior to passing the command string to TSO/E.

[1] Some of these commands may not be supported on all hosts.

By default, the results of a local command get displayed at the user's terminal.

The following is an example of the **LOCAL** command being used within IBM z/OS NetEx/eFT to obtain a listing of local files or data sets having the default TSO/E prefix "GUEST1". The command executed is the TSO/E command "LISTCAT":

```
eFT> local listcat
IN CATALOG:CAT.MVS3.TSO
GUEST1.BATCH.DATA
GUEST1.BATCH.JCL
GUEST1.STAFF.MAIL.DOC
GUEST1.MYMVS.PDS
GUEST1.NEW.LETTER.TXT
GUEST1.SCRIPT.UA
```

Using **LOCAL** from within a NetEx/eFT session, it is also possible to invoke a compiler, execute a CLIST or user written application, submit JCL, etc. For example, to submit a previously written JCL batch stream called "BATCH.JCL", the TSO/E "SUBMIT" command is used:

```
eFT> local submit batch.jcl
JOB GUEST1T(JOB04650) SUBMITTED
```

Now to check on the status of the job, the TSO/E "STATUS" command can be issued:

```
eFT> local status
JOB GUEST1T(JOB04650) WAITING FOR EXECUTION
```

IBM z/OS NetEx/eFT **LOCAL** command execution also supports the ISPF "ISPEXEC" command interface on ISPF version 2.3 or higher. For this to take place, NetEx/eFT must be invoked as an ISPF application. By default, the NetEx/eFT Initiator (client) CLIST will invoke the NetEx/eFT client as an ISPF application if the user is running under a valid ISPF environment.

Some versions of NetEx/eFT support a **LOCAL INTERACTIVE** qualifier that allow the user to run interactively with local applications. Under IBM z/OS NetEx/eFT, local command execution is always in an interactive mode. In fact, by leaving the command parameter off when issuing the **LOCAL** command, the user can enter an interactive TSO/E session. For example:

```
eFT> local
Type 'EXIT' to return to USER prompt

USER READY
```

At this point the user can interact with TSO/E in the usual manner. To return to the NetEx/eFT session, the user must exit TSO/E by typing "EXIT":

```
USER READY
exit

eFT>
```

"EXIT" returns the user back to the NetEx/eFT session where all remote connections, alias definitions, and the like have been retained. Local interactive mode makes it easy for a user to bring up NetEx/eFT, establish a remote connection, and then return to TSO/E for further z/OS activity. When a file or remote job is requested, the user simply returns to NetEx/eFT where the remote host is actively waiting.

## LOCAL Command Status

The completion status of the **LOCAL** TSO/E command is placed in the NetEx/eFT variable **{status:local}**. The possible values are:

| | |
|---|---|
| **0** | Normal (successful) completion. |
| **N** | Integer value for unsuccessful completion. Set by TSO/E command. |
| **Sxxx** | Abnormal completion with z/OS system ABEND xxx. |
| **Uxxxx** | Abnormal completion with user ABEND xxxx. |
| **Interrupt** | Unsuccessful completion due to a keyboard interrupt (attention). |
| **Error** | Unsuccessful completion due to invalid, unknown, or unsupported TSO/E command. |

For more information on the **LOCAL** command and its qualifiers, refer to "Command Descriptions" on page 117.

# Issuing Local IBM z/OS Host-Independent Commands

As on the remote host, the local NetEx/eFT user has the option of executing native host commands, host independent commands, or user defined aliases. The host independent commands allow a user to execute commands on many different systems with a single command set. To display the list of host independent commands defined for IBM z/OS NetEx/eFT, issue the **SHOW LOCAL ALIAS** command:

```
eFT> show local alias
eFT:
eFT: BRowse ............ {dfn(1,"ISPEXEC BROWSE DATASET({1})", "ISPEXEC SEL
eFT: ECT PGM(ISRBRO) PARM(ISRBRO01)")}
eFT: CANcel ............ CANCEL
eFT: COPY .............. SMCOPY FROMDATASET({1}) TODATASET({2}) NOTRANS
eFT: DELete ............ DELETE
eFT: DIFference ........ (TYPE)DIFFERENCE is not available for MVS
eFT: DIRectory ......... LISTCAT
eFT: EDit .............. {dfn(1,"ISPEXEC EDIT DATASET({1})", "ISPEXEC SELEC
eFT: T PGM(ISREDIT) PARM(P,ISREDM01)")}
eFT: HELP .............. HELP
eFT: ISPF .............. ISPEXEC SELECT PANEL(ISR@PRIM) OPT({1}) NEWAPPL(IS
eFT: R) NEWPOOL
eFT: MEMbers ........... LISTDS {dfn(1,"{1} MEMBERS","'{directory:local}' L
eFT: EVEL")} HISTORY
eFT: PRInt ............. PRINTDS DATASET({1}) {2} {3} {4} {5}
eFT: QUEue ............. STATUS
eFT: REName ............ RENAME
eFT: SUBmit ............ SUBMIT
eFT: TYPe .............. %EFTTYPE
eFT: WHO ............... (TYPE)WHO is not available for MVS
eFT:
```

The host independent commands are in the left column and the IBM z/OS TSO/E command translations are in the right column. Users can issue host independent commands as if they were commands native to TSO/E. NetEx/eFT takes care of the translation. Notice that the commands **DIFFERENCE** and **WHO** are simply mapped to a special keyword **(TYPE)** with the message "xxx is not available for z/OS". If the user was to invoke one of these commands, NetEx/eFT, upon seeing **(TYPE)**, would display the message. These host independent commands have no equivalent TSO/E translation.

Below is a list of all standard z/OS host independent commands along with a description of how they are used. From a local NetEx/eFT Initiator, any of these commands can be invoked on a remote z/OS system by means of the **LOCAL** command.

**BRowse** This alias can be used if eFT was invoked from ISPF. This alias will ISPF "BROWSE" the remote file specified.

**CANcel** cancel or halt processing of a previously submitted batch job. The user must have authorization from installation management to use "CANCEL". Use the **QUEUE** alias to display the status of batch jobs. **CANCEL** is mapped to the TSO/E "CANCEL" command. The format is:

```
CANCEL jobname[(jobid)]
```

**COPY** copy a z/OS data set (file) to another data set (file). The source data set must be a sequential data set or a member of a partitioned data set with either fixed or variable length records. The destination data set, if it exists, must also be sequential or a member of a partitioned data set with either fixed or variable length records. **COPY** is mapped to the TSO/E "SMCOPY" command. The format is:

```
COPY src_data_set[(member)] dest_data_set[(member)]
```

**DELete** delete one or more data sets or a member of a partitioned data set on the z/OS host. An asterisk can be inserted into the data set name to select entries having similar names. Only one asterisk per data set name may be used and it cannot appear in the first position. **DELETE** is mapped to the TSO/E "DELETE" command. The format of **DELETE** is:

```
DELETE data_set[(member)]
```

**DIFference** There is no support for this command under z/OS TSO/E.

**DIRectory** display a listing of data sets from a catalog. By default, **DIRECTORY**, lists the data sets with the current TSO/E prefix. **DIRECTORY** is mapped to the TSO/E "LISTCAT" command. The format is:

```
DIRECTORY [LEVEL(data_set_level)]
```

**EDIT** can be used if eFT was invoked from ISPF. This alias will ISPF "EDIT" the remote file specified.

**HELP** obtain help on a z/OS TSO/E function, command, subcommand, etc. **HELP** is mapped to the TSO/E "HELP" command. The format is:

```
HELP [TSO/E_topic]
```

**ISPF** is not for general use. This alias exists to facilitate a separate eFT feature.

**MEMbers** used to obtain a list of members in a remote z/OS PDS or PDSe by using the TSO "LISTDS" command. The format is:

```
MEMBERS pds_dataset_name
```

**PRInt** format and print a data set on any printer defined to the z/OS Job Entry System (JES). **PRINT** is mapped to the TSO/E "PRINTDS" command. The format is:

```
PRINT data_set[(member)] [CLASS(class)][DEST(destination)]
```

**QUEue** display the current status of the user's batch jobs. **QUEUE** is mapped to the TSO/E "STATUS" command. The format is:

```
QUEUE [jobname[(jobid)]]
```

**RENname** change the name of a cataloged data set or member of a partitioned data set. **RENAME** is mapped to the TSO/E "RENAME" command. The format is:

```
RENAME old_data_set[(oldmem)] new_data_set[(newmem)]
```

If a member name is used, the "old_data_set" and the "new_data_set" must be the same.

**STAtus** generally, the host independent command **STATUS** is used to display current activity on a particular host. Since TSO/E already supports a "STATUS" command (which displays batch job status), this command is not redefined by NetEx/eFT.

**SUBmit** submit an existing z/OS data set for background (batch job) processing. Users must be authorized by installation management to use **SUBMIT**. **SUBMIT** is mapped to the TSO/E "SUBMIT" command. The format is:

```
SUBMIT data_set[(member)]
```

**TYPe** type out the contents of an existing z/OS data set or member of a partitioned data set. **TYPE** is mapped to a TSO/E CLIST called "%EFTTYPE". The format is:

```
TYPE data_set[(member)]
```

**WHO** There is no support for this command under TSO/E.

The following is an example of a **LOCAL** command that invokes a host independent command called **QUEUE**. **QUEUE** translates to the TSO/E command "STATUS" which gives the status of the user's batch jobs in the queue:

```
eFT> local queue
JOB SMITHX1T(JOB14446) ON OUTPUT QUEUE
JOB SMITHX1T(JOB14447) WAITING FOR EXECUTION
```

Notice that the output from **QUEUE** is equivalent to the output from the TSO/E command "STATUS":

```
eFT> local status
JOB SMITHX1T(JOB14446) ON OUTPUT QUEUE
JOB SMITHX1T(JOB14447) WAITING FOR EXECUTION
```

Local IBM z/OS NetEx/eFT users can also create their own local aliases using the **SET LOCAL ALIAS** command. For example, to create a local alias called "HC" that gives help for the TSO/E "CALL" command (i.e., "HELP CALL"), issue the following command:

```
eFT> set local alias hc help call
```

Now the **SHOW LOCAL ALIAS** command can be used to display the new alias:

```
eFT> show local alias hc
eFT: HC ............... help call
```

This new alias is equivalent to the TSO/E command "HELP CALL" and is stored along with the local host independent commands. Users can create as many local aliases as desired. To make them available for use in all NetEx/eFT sessions, edit them into a local NetEx/eFT startup file.

Refer to "Command Descriptions" on page 117 of this manual for the list of valid **LOCAL** qualifiers and an example of their use. Also see "Advanced Local User's Guide" on page 63 for further discussion on host aliases and aliases in general.

# Editing Remote Files with an Editor

Using a predefined alias called **EDIT**, users can invoke their favorite IBM z/OS editor to edit files that reside on the remote host in which they are connected. The **EDIT** alias is typically defined in the NetEx/eFT site startup file but can easily be redefined and customized in the user's startup file. The following is a sample **EDIT** alias that invokes the IBM z/OS ISPF editor.

```
eFT> show alias edit
eFT: EDit .. receive -create replace -mode character {1} edit.tmp
eFT:         ledit edit.tmp
```

```
eFT:          ask -prompt "Update remote (Y/N)? " -default "Y" yn
eFT:          {cmp("Yes",yn,"send -create replace edit.tmp {1}")}
eFT:          local delete edit.tmp
```

To invoke **EDIT**, the user simply types **EDIT** followed by the name of an existing file on the remote host:

```
eFT> edit rfile
```

In the example, the remote file "rfile" would automatically be transferred to the z/OS system and the ISPF editor would be invoked. The user would then edit the file in the normal way. Once the edit session is over, the user has the option of overwriting the remote file or not. Finally, the local temporary file is deleted. This **EDIT** alias can be greatly enhanced to address file protection, loss of remote connection, etc. It is up to the site to determine exactly how **EDIT** should function in each environment.

Refer to the "Advanced Local User's Guide" section on page 63 for more information on developing multi-command aliases.

# Interrupting a Command within IBM z/OS NetEx/eFT

To interrupt or terminate a command from executing once it has started, press the attention key (or <PA1>). By intercepting TSO/E attention key processing, NetEx/eFT aborts any NetEx/eFT, local, or remote host command within a few seconds. Often it is desirable to interrupt a command if the output becomes too lengthy (e.g., a directory listing), or if the operation is no longer wanted (e.g., sending a group of files). Interrupting a command with a single attention key strike will result in the NetEx/eFT prompt being displayed unless an alias or input script is handling interrupts specially. (This is discussed in more detail in the "Advanced Local User's Guide" section on page 63.)

To terminate a NetEx/eFT session that appears to have stopped for some reason, generate three attention interrupts in rapid succession. Three consecutive interrupts cause IBM z/OS NetEx/eFT to abort with a "U0002" ABEND code.

# Using Special Characters with Other Hosts

When connected to an OpenVMS host through NetEx/eFT, it is recommended that one uses the left- and right-angle brackets (<, >) in place of the normal square brackets typically used in VMS file specification strings. For example, to change the **REMOTE DIRECTORY** on a VMS host to "[ROOT.SOURCE]", type:

```
eFT> set remote directory <ROOT.SOURCE>
```

Note that square brackets are often not defined in an IBM terminal's character set.

The TSO/E "TERMINAL" command can also be used to convert nondisplay characters to display characters during a TSO/E session. The "EFTUSER" CLIST provided with z/OS NetEx/eFT contains an example of using the TSO/E TERMINAL command.

# IBM z/OS Remote User's Guide

This section is intended for users who are currently running the NetEx/eFT Initiator from any local host, regardless of operating system, and would like to establish a network connection into an IBM z/OS host. The information provided here and in "File Handling Under IBM z/OS NetEx/eFT", should be enough to help a non-z/OS user start being productive in a very short period of time. z/OS users should also reference these sections to become comfortable with how NetEx/eFT operates in the environment. The IBM z/OS Time Sharing Option (TSO/E) environment is presented to the remote user for interaction.

## Connecting to an IBM z/OS Host

A NetEx/eFT network connection is established into an IBM z/OS environment by means of a process known as the Multiplex Server (MUX Server). The task of the MUX Server is to listen on the network for remote connect/login requests. Upon receiving a connect request, the MUX Server starts a VTAM session and issues a **LOGON** command to TSO/E. TSO/E then validates the user ID and password to ensure that proper security is checked. Finally, the MUX Server, after monitoring a successful logon through TSO/E, starts up a NetEx/eFT Helper program under the TSO/E session. The Helper and remote Initiator then exchange connection information through the MUX Server, and network connection is established.

As seen above, connections to an IBM z/OS host typically requires a TSO/E username and password. For example, if a username on host MVS003 is "SMITH" with a password of "JOHN", type:

```
eFT> connect mvs003 smith john
```

Some IBM z/OS hosts with additional security packages (such as RACF) may require periodic changing of a user's password. This can be accomplished in NetEx/eFT by using the TSO/E syntax *password/newpassword*. For example, to change the password in the previous example from "JOHN" to "JACK", type:

```
eFT> connect mvs003 smith john/jack
```

Additional parameters to the TSO/E logon command can be provided in the **CONNECT** command following the password. For example, if one does not want to receive mail messages during logon processing ("NOMAIL"), type:

```
eFT> connect mvs003 smith john nomail
```

If multiple TSO/E logon parameters are required, they must be enclosed in double quotes following the password. For example, to initiate a session with a different logon procedure ("PROC") as well as disabling mail, type:

```
eFT> connect mvs003 smith john "proc(MYJCL) nomail"
```

The values for the **CONNECT** qualifiers **ACCOUNT** and **PROFILE** are mapped to the TSO/E "LOGON" command keywords "ACCT" and "PROC". A user can use **SET CONNECT** to establish default values or the NetEx/eFT responder on the IBM z/OS host can provide defaults. The following **CONNECT** command is equivalent to the previous example:

```
eFT> connect mvs003 smith john -profile MYJCL nomail
```

Some TSO/E logon procedures may prompt for additional information (e.g., security or accounting information). Responses to these prompts can be provided following the TSO/E logon parameters (or a null string if no logon parameters are required). For example, if the normal TSO/E logon procedure prompts for a department code "ENG01" and a charge number "7743", type:

```
eFT> connect mvs003 smith john " " ENG01 7743
```

Finally, disconnecting from an IBM z/OS host (using **DISCONNECT**, **EXIT**, or **QUIT**) will cause an automatic TSO/E logoff.

# CONNECT Qualifiers Used by IBM z/OS NetEx/eFT

The **CONNECT** command (which is used by the **LOGIN** alias) has several qualifiers associated with it, of which some are only used by certain systems to assist in the login process. Some **CONNECT** qualifiers of special importance to IBM z/OS are:

**-ACCount** Passed to an IBM z/OS host as the TSO/E logon parameter "ACCT(xxx)". The string value passed to **ACCOUNT** should be only the "xxx" parameter within the parenthesis.

**-PROFile** Passed to an IBM z/OS host as the TSO/E logon parameter "PROC(xxx)". The string value passed to **PROFILE** should be only the "xxx" parameter within the parenthesis.

The following **CONNECT** qualifiers are normally ignored by IBM z/OS NetEx/eFT at connect/login time:

APPlication
COMmand
PROJect
SCRIpt
SECondary
SECURE
SITE

**Note:** Your site may have customized IBM z/OS NetEx/eFT to use one or more of these additional qualifiers.

The **CONNECT** command as described in "Command Descriptions" of the local NetEx/eFT guide lists all applicable **CONNECT** qualifiers.

# Remote IBM z/OS NetEx/eFT Startup Files

After establishing a successful connection into a remote z/OS host, NetEx/eFT executes the startup files as described by the **CONNECT SEARCH** qualifier on the Initiator. The normal default definition of this qualifier is:

```
eFT> show connect search
eFT: SEArch ............ (SITE) (USER)
```

Possible values for **CONNECT SEARCH** are the keywords **(NONE)**, **(SITE)**, **(USER)**, or any valid remote file specification containing NetEx/eFT commands. The definitions for the special keywords as they relate to NetEx/eFT for IBM z/OS are given below.

**(NONE)** do not process any responder (or server) startup files.

**(SITE)** implies a NetEx/eFT site startup file (partitioned data set member) called "SSERVER" located in the remote IBM z/OS site partitioned data set. This is explained in more detail in "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106.

**(USER)** implies a NetEx/eFT user-level startup file (sequential dataset) called "SERVER.UA" that is located in the user's login TSO/E prefix. This is explained in more detail in "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106.

No server startup files are required. If any do exist, NetEx/eFT sends their contents (a sequence of NetEx/eFT commands) back to the local Initiator where they are then executed in the order described by the **SEARCH** qualifier. The following commands may not be executed from a server startup file:

CONNECT
DISCONNECT
LOCAL
RECEIVE
REMOTE
SEND

# Transferring Files to an IBM z/OS Host

The file transfer capabilities of NetEx/eFT are provided by two commands, **SEND** and **RECEIVE**. The **SEND** command provides file transfer from the local host to a remote IBM z/OS host. The **RECEIVE** command transfers files from a remote IBM z/OS host back to the local host.

The **SEND** and **RECEIVE** commands function the same regardless of the host from which they are executed. However, the command qualifiers to **SEND** and **RECEIVE** differ depending on the hosts involved. The qualifiers affect how files are stored, transferred, named, etc. For details on the **SEND** and **RECEIVE** qualifiers that exist for file transfers to and from an IBM z/OS system, refer to "File Handling Under IBM z/OS NetEx/eFT" on page 41.

# Executing Remote TSO/E Commands

The **REMOTE** command gives users the ability to execute TSO/E commands on a remote IBM z/OS system and view the results. Host commands can either be a native TSO/E command, a TSO/E CLIST, or a remote alias (or host-independent command) having a valid TSO/E command translation.

The following is an example of the **REMOTE** command being used within IBM z/OS NetEx/eFT to check the status of the user's batch jobs previously queued. The TSO/E command is STATUS:

```
eFT> remote status
eFT:  JOB SMITHX1T(JOB14446) ON OUTPUT QUEUE
eFT:  JOB SMITHX1T(JOB14447) WAITING FOR EXECUTION
```

The prefix **eFT:** indicates that the results are being returned from the IBM z/OS host. This prefix can be modified or turned off using the **SET REMOTE PREFIX** command.

Using **REMOTE** to execute commands on an IBM z/OS host, it is possible to do tasks such as submit a batch job, invoke a compiler, execute a CLIST, status a queue, delete a file, etc. Any non-interactive, non-full-screen oriented TSO/E command can be issued. Interactive commands that require users to respond to prompts or full screen-oriented applications cannot be run through NetEx/eFT using the **REMOTE** command. It should be noted, however, that many interactive tasks can still be performed by providing a CLIST containing the requested information.

## REMOTE Command Status

The completion status of the **REMOTE** TSO/E command is placed in the NetEx/eFT variable **{status:remote}**. The possible values are:

| | |
|---|---|
| **0** | Normal (successful) completion. |
| **N** | Integer value for unsuccessful completion. Set by TSO/E command. |
| **Sxxx** | Abnormal completion with z/OS system ABEND xxx. |
| **Uxxxx** | Abnormal completion with z/OS system ABEND xxxx. |
| **Interrupt** | Unsuccessful completion due to a keyboard interrupt (attention). |
| **Error** | Unsuccessful completion due to invalid, unknown, or unsupported TSO/E command. |

For more information on the REMOTE command, refer to “Command Descriptions” on page 117 of this manual. For more information on the TSO/E command set, refer to the IBM documentation.

# Issuing Remote IBM z/OS Host Independent Commands

Although executing TSO/E commands from a remote system may be very useful, many times remote users are not familiar with the TSO/E command set. Therefore, NetEx/eFT makes a set of Host Independent Commands available for all users around the network to use, without requiring them to learn each host’s command set. These commands are implemented as remote aliases. To display the list of host independent commands defined for IBM z/OS NetEx/eFT, issue the **SHOW REMOTE ALIAS** command:

```
eFT> show remote alias
eFT:
eFT: CANcel ............ CANCEL
eFT: COPY .............. SMCOPY FROMDATASET({1}) TODATASET({2}) NOTRANS
eFT: DELete ............ DELETE
eFT: DIFference ........ (TYPE)DIFFERENCE is not available for MVS
eFT: DIRectory ......... LISTCAT
eFT: HELP .............. HELP
eFT: MEMbers ........... LISTDS {dfn(1,"{1} MEMBERS","'{directory:remote}'
eFT: LEVEL")} HISTORY
eFT: PRInt ............. PRINTDS DATASET({1}) {2} {3} {4} {5}
eFT: QUEue ............. STATUS
eFT: REName ............ RENAME
eFT: SUBmit ............ SUBMIT
eFT: TYPe .............. %EFTTYPE
eFT: WHO ............... (TYPE)WHO is not available for MVS
eFT:
```

The host independent commands are in the left column and the TSO/E command translations are in the right column. Users can issue host independent commands as if they were commands native to TSO/E. NetEx/eFT takes care of the translation. Notice that a couple of the commands, **DIFFERENCE** and **WHO**, are simply mapped to a special keyword **(TYPE)** with the message “xxx is not available for z/OS”. If the user was to invoke one of these commands, NetEx/eFT, upon seeing **(TYPE)**, would display the message. These host independent commands have no equivalent TSO/E translation.

Below is a list of all standard z/OS host independent commands along with a description of how they are used. From a local NetEx/eFT Initiator, any of these commands can be invoked on a remote z/OS system by means of the **REMOTE** command.

**CANcel** cancel or halt processing of a previously submitted batch job. The user must have authorization from installation management to use **CANCEL**. Use the **QUEUE** alias to display the status of batch jobs. **CANCEL** is mapped to the TSO/E “CANCEL” command. The format is:

```
CANCEL jobname[(jobid)]
```

**COPY** copy a z/OS data set (file) to another data set (file). The source data set must be a sequential data set or a member of a partitioned data set with either fixed or variable length records. The destination data set, if it exists, must also be sequential or a member of a partitioned data set with either fixed or variable length records. **COPY** is mapped to the TSO/E “SMCOPY” command. The format is:

```
COPY src_data_set[(member)] dest_data_set[(member)]
```

**DELete** delete one or more data sets or a member of a partitioned data set on the z/OS host. An asterisk can be inserted into the data set name to select entries having similar names. Only one asterisk

per data set name may be used and it cannot appear in the first position. **DELETE** is mapped to the TSO/E "DELETE" command. The format of **DELETE** is:

```
DELETE data_set[(member)]
```

**DIFference** There is no support for this command under z/OS TSO/E.

**DIRectory** display a listing of data sets from a catalog. By default, **DIRECTORY**, lists the data sets with the current TSO/E prefix. **DIRECTORY** is mapped to the TSO/E "LISTCAT" command. The format is:

```
DIRECTORY [LEVEL(data_set_level)]
```

**HELP** obtain help on a z/OS TSO/E function, command, subcommand, etc. **HELP** is mapped to the TSO/E "HELP" command. The format is:

```
HELP [TSO/E_topic]
```

**MEMbers** used to obtain a list of members in a remote z/OS PDS or PDSe.

```
MEMBERS pds_dataset_name
```

**PRInt** format and print a data set on any printer defined to the z/OS Job Entry System (JES). **PRINT** is mapped to the TSO/E "PRINTDS" command. The format is:

```
PRINT data_set[(member)] [CLASS(class)][DEST(destination)]
```

**QUEue** display the current status of the user's batch jobs. **QUEUE** is mapped to the TSO/E "STATUS" command. The format is:

```
QUEUE [jobname[(jobid)]]
```

**REName** change the name of a cataloged data set or member of a partitioned data set. **RENAME** is mapped to the TSO/E "RENAME" command. The format is:

```
RENAME old_data_set[(oldmem)] new_data_set[(newmem)]
```

If a member name is used, the "old_data_set" and the "new_data_set" must be the same.

**STAtus** generally, the host independent command **STATUS** is used to display current activity on a particular host. Since TSO/E already supports a "STATUS" command (which displays batch job status), this command is not redefined by NetEx/eFT.

**SUBmit** submit an existing z/OS data set for background (batch job) processing. Users must be authorized by installation management to use **SUBMIT**. **SUBMIT** is mapped to the TSO/E "SUBMIT" command. The format is:

```
SUBMIT data_set[(member)]
```

**TYPe** type out the contents of an existing z/OS data set or member of a partitioned data set. **TYPE** is mapped to a TSO/E CLIST called "%EFTTYPE". The format is:

```
TYPE data_set[(member)]
```

**WHO** There is no support for this command under TSO/E.

The following is an example of a **REMOTE** command that invokes the host independent command **TYPE**. **TYPE** translates to a TSO/E CLIST called "%EFTTYPE" which types out the contents of a data set (in this case SECTION.ONE), or member of a partitioned data set:

```
eFT> remote type section.one
eFT:
eFT:  ************************************************
eFT:  *   THIS IS THE CONTENTS OF DATA SET SECTION.ONE   *
eFT:  ************************************************
eFT:    SECTION ONE:
eFT:
eFT:       1) The Command Translation
eFT:       2) The Command Execution
eFT:       3) The Command Termination
eFT:       4) The Command Status Checking
eFT:
eFT:    END OF SECTION ONE.
eFT:
```

The output from **TYPE** is equivalent to the output that would be seen from the TSO/E "%EFTTYPE" CLIST.

Users can also create their own remote aliases using the **SET REMOTE ALIAS** command. For example, to create a remote alias called **FILES** that gives a listing of the user's data sets (i.e., TSO/E "LISTCAT" command), issue the following command:

```
eFT> set remote alias files listcat
```

Now the **SHOW REMOTE ALIAS** command can be used to display the new alias:

```
eFT> show remote alias files
eFT: FILES ............... LISTCAT
```

This new alias is equivalent to the **DIRECTORY** host independent command and is stored in the same fashion. Users can create as many remote aliases as desired. To make them available for use in all NetEx/eFT sessions, edit them into a remote NetEx/eFT startup file.

Refer to the "Advanced Local User's Guide" on page 63 for further discussion on host aliases and aliases in general.

# File Handling Under IBM z/OS NetEx/eFT

This section is intended to address IBM z/OS file handling issues as they relate to the NetEx/eFT file transfer commands **SEND** and **RECEIVE**. Since files on an IBM z/OS system are usually referred to as data sets, these terms will be used interchangeably throughout this section. Both local and remote z/OS NetEx/eFT users should use this section as a reference for transferring files to and from a z/OS host. Prior to reading this section, it is important to understand the following terminology:

**Source Host** Refers to the host in which the source file (of either a **SEND** or **RECEIVE**), resides. The source file is the existing file which is being transferred to the destination host.

**Destination Host** Refers to the host in which the destination file (of either a **SEND** or **RECEIVE**), will be created or overwritten. The destination file is the new or replaced file that results following a file transfer.

The distinction between Source Host and Destination Host is important since both the **SEND** and **RECEIVE** commands always transfer files from the Source Host to the Destination Host. **SEND** and **RECEIVE** command qualifiers are usually tied directly to the Destination Host since that is where files get created.

The following section describes the qualifiers for the **SEND** and **RECEIVE** commands that exist for file transfers when an IBM z/OS system is the Destination Host.

## IBM z/OS File Transfer Qualifiers and Default Values

Below is a list of the **SEND** and **RECEIVE** command qualifiers that are available for file transfers when an IBM z/OS system is the destination host. That is, when the local host is a z/OS system, the qualifiers listed below pertain to the **RECEIVE** command (local z/OS is the Destination Host). When the remote host is a z/OS system, the qualifiers listed below are valid for the **SEND** command (remote z/OS is the Destination Host).

**Table 1. z/OS File Transfer Qualifiers and Default Values**

| Qualifier | Description |
|---|---|
| **-BLOCKsize** (INTEGER) | This qualifier specifies the maximum length in bytes per block of the data set to be created. The value for **BLOCKsize** depends on **RECFORmat** and **RECLENgth** but can never be outside the range 0 to 32760.<br><br>This qualifier is only used when creating a new data set. The default **BLOCKsize** is 3120 for **CHARACTER** and **RECORD** modes, and 4104 for **STREAM** mode.<br><br>The minimum spelling is -BLOCK.<br><br>Please check zOS DFSMS documentation. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-CHEcksum** (BOOLEAN) | This qualifier controls if NetEx/eFT should perform a checksum as part of the file transfer. This provides an alternative to the full Cyclic Redundancy Check (CRC) support provided by the **-CRC** qualifier. This optimized checksum algorithm provides additional CPU utilization savings over the optimized CRC algorithm.<br><br>If both **CHEcksum** and **CRC** are enabled, only CRC processing will be performed.<br><br>The default value is OFF.<br><br>The minimum spelling is -CHE. |
| **-COMPress** (BOOLEAN) | This qualifier specifies whether data compression should be performed during file transfer. Refer to the **METHod** qualifier for available compression types.<br><br>The default is OFF.<br><br>The minimum spelling is -COMP. |
| **-CRC** (BOOLEAN) | This qualifier controls if NetEx/eFT should perform a checksum as part of the file transfer. When **CRC** is enabled a 32-bit CRC is calculated by the source host along with a block sequence number. These are verified by the destination host.<br><br>If both **CHECKsum** and **CRC** are enabled, only CRC processing will be performed.<br><br>The default value is OFF. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-CREate** (STRING) | This qualifier indicates to the destination z/OS host how to create the new data set. The available options are:<br><br>NEW - which means if a data set or member of a data set by the same name already exists, do not overwrite it.<br><br>REPLACE - will overwrite an existing data set or member of the same name or create a new one.<br><br>DELETE - similar to REPLACE. But it will delete an existing data set, then create a new one using the file transfer qualifiers to generate new data set attributes.<br><br>APPEND - appends the source file to the destination data set if it exists, otherwise creates a new one.<br><br>DELETE and APPEND are only valid when writing to a sequential data set.<br><br>The default is NEW.<br><br>The minimum spelling is -CRE.<br><br>When **TAPE** is specified and a **VOLSER** is specified then **-CREATE replace** must be specified. Otherwise, when a **VOLSER** is not specified then **-CREATE new**, **-CREATE replace**, or **-CREATE delete** can be specified. |
| **-DDname** (STRING) | This qualifier causes the NetEx/eFT file handling routines to use the provided Data Definition (DD) name instead of requesting MVS to generate a unique one.<br><br>The minimum spelling is -DD. |
| **-DELete_on_error** (BOOLEAN) | This qualifier causes a destination data set to be deleted automatically if the file transfer ends with an error.<br><br>The default is OFF.<br><br>The minimum spelling is -DEL. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-DEVICE_Wait** (INTEGER) | This qualifier specifies the number of seconds that a **SEND** or **RECEIVE** command will wait for a device during data set allocation. The failing allocation request is retried every five seconds during the **DEVICE_Wait** time period. This qualifier is usually used to wait for a tape drive to become available.<br><br>The minimum spelling is -DEVICE_W |
| **-DISPosition** (STRING) | This qualifier tells the destination host what to do with a new data set after successfully allocating it. **DISPosition** can either be CATALOG or KEEP.<br><br>CATALOG says to place an entry pointing to the data set in the system or user catalog.<br><br>KEEP indicates that the data set is to be kept on the volume without cataloging the data sets.<br><br>DISPOSITION is only used when creating a new data set in which case CATALOG is the default. Existing data sets always get allocated with a disposition of (OLD,KEEP) regardless of the DISPOSITION qualifier.<br><br>The default is CATALOG.<br><br>The minimum spelling is -DISP. |
| **-EXPand** (BOOLEAN) | This qualifier specifies whether data that was transferred with the **-COMPress** qualifier should be expanded (decompressed) on the receiving host. Must be used in conjunction with the **-COMPress** qualifier.<br><br>The default value is OFF.<br><br>The minimum spelling is -EXP. |
| **-EXPIRation** (STRING) | This qualifier specifies the expiration date in "YYYYDDD" or "YYDDD" format. This qualifier is the same as the z/OS JCL "LABEL=" keyword "EXPDT=" operand and may also be used for DASD data sets. Special expiration dates such as "98000" are supported. This qualifier is mutually exclusive with the **-RETENtion** qualifier.<br><br>The minimum spelling is -EXPIR. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-FLOW** (BOOLEAN) | This qualifier controls if NetEx/eFT enables file transfer flow control. When **FLOW** is on, every block to be transferred must be requested by the receiving host. The sender sends a block only when the receiver is ready for one. **FLOW** exists to prevent unusual read timeouts during file transfers that can be caused, for example, by an interactive/selective restore from an archive file. Waiting for an operator to load a tape is another example of when **FLOW** may be required. Because each block must be requested by the NetEx/eFT receiver, a significant penalty in performance is paid when FLOW is enabled.<br><br>The default is OFF. |
| **-KEEPBLanks** (BOOLEAN) | This qualifier allows the user to turn on or off automatic deletion of blank (EBCDIC x'40') characters from the end of fixed length records. This qualifier only applied to MVS data sets with fixed length records which are transferred using **-MODE CHARACTER** or **-MODE V1CHAR**.<br><br>The default is OFF.<br><br>The minimum spelling is -KEEPBL. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-LABELnumber** (INTEGER) | This qualifier specifies the relative file sequence number. The first data set on a set of tape volumes is file sequence number "1", the second data set is "2", etc. This qualifier is equivalent to the first positional parameter of the z/OS JCL "LABEL=" keyword. NetEx/eFT automatically retains volume serial information between tape data sets. This means that after creating tape data set one, the data set name is retained, so that when tape data set two is created it will be started on the last volume used by data set one. The accepted range of values is between 0 and 9999.<br><br>Specifying a value of zero causes NetEx/eFT to increment the last supplied **-LABELnumber** value by one and use that value for the new file sequence number. If no value was previously supplied, the new file sequence number is set to one.<br><br>z/OS does not currently support retaining a volume allocated with Dynamic Allocation (SVC 99) between data sets. This means that the last volume used by data set one will be physically dismounted after data set one is closed and then remounted if data set two is opened. The data sets are created properly but creating multiple data set tape volumes will generate a tape dismount and remount between each data set. This is an MVS limitation, not a NetEx/eFT limitation. It is intended for NetEx/eFT to support volume retention as soon as z/OS Dynamic Allocation does.<br><br>If volume dismounts are not acceptable at your site, MVS JCL can be created supplying the "VOL=(,RETAIN)" keyword and NetEx/eFT file transfer can be performed using the "DD:xxxxxxxx" facility. In this case, volume retention is performed as specified in the MVS JCL User's Guide.<br><br>The default is "".<br><br>The minimum spelling is -LABEL. |
| **-LABELTYPE** (STRING) | This qualifier specifies the label type to be used for a tape data set. Valid values are SL (standard label), AL (ANSI label), and NL (no label). If a value of AL or NL is specified, the **LABELnumber** qualifier must also be specified. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-MAXRECord** (INTEGER) | This qualifier specifies the maximum allowed record size when reading files on the source host. **MAXRECord** can only be used when the current transfer mode is RECORD mode. If an attempt is made to transfer a file that has records larger than **MAXRECord**, the transfer will terminate with an appropriate error message. The accepted range of values is 1-65535.<br><br>The default is 10000.<br><br>The minimum spelling is -MAXREC. |
| **-METHod** (STRING) | This qualifier specifies the data compression method. It must be used in conjunction with the **-COMPress** and/or **-EXPand** qualifiers.<br><br>This value can be set to either LZW or RLE. LZW specifies Lempel-Ziv-Welch compression, and RLE specifies run-length-encoding compression. The default method is LZW.<br><br>The minimum spelling is -METH. |
| **-MODe** (STRING) | This qualifier specifies the current file transfer mode. NetEx/eFT for IBM z/OS supports the following modes: CHARACTER, STREAM, RECORD, BACKUP, RESTORE, V1CHAR, and COPY. The value of MODE is used internally by NetEx/eFT to decide how to access or create the files being transferred. A mode must be supported by both hosts in order to have a successful transfer. For more information on transfer modes, refer to “Transfer Modes Supported Under IBM z/OS NetEx/eFT” on page 58.<br><br>The default is CHARACTER.<br><br>The minimum spelling is -MOD. |
| **-PARTialrecord** (BOOLEAN) | This qualifier specifies that records of length greater than the network block size can be transferred.<br><br>If **-PARTialrecord** is OFF, - **MAXRECord** must be less than or equal to the network/NetEx block size.<br><br>If **-PARTialrecord** is ON, **MAXRECord** may exceed the network/NetEx block size.<br><br>The default is ON. The minimum spelling is -PART. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-PDSE** (BOOLEAN) | This qualifier allows a new data set to be created as a PDS/E instead of a PDS. This qualifier is ignored for sequential data sets.<br><br>The default is OFF. |
| **-QUIet** (BOOLEAN) | This qualifier tells NetEx/eFT whether or not to display informational messages during file transfer. This value should be set to either ON or OFF.<br><br>The default is OFF.<br><br>The minimum spelling is -QUI. |
| **-RECALL** (BOOLEAN) | This qualifier indicates whether or not to wait for the recall of a migrated or archived data set before sending the data set. A destination data set is always recalled before overwriting.<br><br>The default is OFF. |
| **-RECFORmat** (STRING) | This qualifier specifies the format and the characteristics of the records in a data set to be created. A user can create a data set with a format of fixed (F), variable (V), or undefined (U). One can also add characteristics such as blocked (e.g., FB), spanned (e.g., VS), blocked with ANSI control characters (e.g., FBA), or blocked with machine code control characters (e.g., VBM).<br><br>The default is variable blocked (VB).<br><br>The minimum spelling is -RECFOR. |
| **-RECLENgth** (INTEGER) | This qualifier specifies: (1) the length in bytes for fixed-length records, (2) the maximum length in bytes for variable-length records, or (3) the maximum length in bytes for undefined length blocks of the data set to be created.<br><br>For variable-length records, **RECLENgth** is the number of bytes of user data per record and does not include the overhead length bytes. A new variable length data set is allocated with **RECLENgth** +4 bytes to allow for length byte overhead. This value must make sense for the current **RECFORmat** and **BLOCKsize** and can never be outside the range 1 to 32760.<br><br>The default is 251 for CHARACTER and RECORD mode transfers, and 4096 for STREAM mode transfers.<br><br>The minimum spelling is -RECLEN. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-REFER** (STRING) | This qualifier can be used in one of two ways. First, to cause the next output data set to be placed on the same volume serial as the data set name provided via the **-REFER** qualifier. The **-REFER** qualifier can provide this function for either DASD or tape. Second, the **-REFER** qualifier with the special data set name of '*' can be used in conjunction with the **-DDname** and **-RETAIN** qualifiers to cause the next output data set to be placed immediately following the last output data set associated with the **-DDname** value. This operation occurs without an intervening tape remount, enabling tape stacking "refer back" via Dynamic Allocation. This is similar to what can be achieved with JCL and the VOL=(,RETAIN,REF=*.xxx.xxx) keywords. |
| **-RELEase** (BOOLEAN) | This qualifier requests that space allocated to a new sequential data set, but not used, is to be released when the data set is closed. It is ignored for partitioned data sets.<br><br>The default for **RELEase** is ON.<br><br>The minimum spelling is -RELE |
| **-RETAIN** (BOOLEAN) | This qualifier causes the NetEx/eFT file handling routines to skip the dynamic deallocation request usually performed after the data set has been closed.<br><br>Using **-RETAIN** requires that the **-DDname** qualifier be provided, since the DD name will not be unallocated at the end of the file output operation. This allows the DD name to be reused by subsequent output requests.<br><br>It is important to remember that the user is now responsible for causing the deallocation to occur, either by performing another output operation without the **-RETAIN** qualifier set, or by explicitly issuing a TSO/E FREE command. If neither of these two items happens, the DD name and device(s) associated with it remain allocated until the job step terminates, which in the case of an interactive TSO/E session may be for a longer period of time than is desirable.<br><br>It is important to remember that the user is now responsible for causing the deallocation to occur, either by performing another output operation without the **-RETAIN** qualifier set, or by explicitly issuing a TSO/E FREE command. If neither of these two items happens, the DD name and device(s) associated with it remain allocated until the job step terminates, which in the case of an interactive TSO/E session may be for a longer period of time than is desirable. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-RETENtion** (INTEGER) | This qualifier specifies the retention period in days. This qualifier is the same as the MVS JCL "LABEL=" keyword "RETPD=" operand and may also be used for DASD data sets. This qualifier is mutually exclusive with the **-EXPIRation** qualifier.<br><br>The minimum spelling is -RETEN. |
| **-RNT** (BOOLEAN) | This qualifier indicates whether or not the Resilient Network Transfer (RNT) option is enabled for this file transfer. This option provides the ability for the sender to resume a file transfer operation from the point of interruption. If **RNT** is enabled, the network connection between a NetEx/eFT client/server pair is reestablished following a network outage and the file transfer operation is resumed. This value should be set to either ON or OFF.<br><br>The default is OFF. |
| **-RNT_BUFalloc** (INTEGER) | This qualifier specifies the size of the RNT buffer. This parameter indicates the number of bytes retained by the sending host, in order to reestablish a network connection and restart a file transfer from the point of interruption. The accepted range of values is 1 to 1,048,576 bytes.<br><br>The default is 262144 bytes.<br><br>The minimum spelling is -RNT_BUF. |
| **-RNT_INTerval** (INTEGER) | This qualifier specifies the number of seconds between attempts to re-establish the NetEx/eFT session during RNT processing. The accepted range of values is 1 to 600 seconds.<br><br>The default is 60 seconds.<br><br>The minimum spelling is -RNT_INT |
| **-RNT_TIMeout** (INTEGER) | This qualifier specifies the number of seconds to wait before abandoning attempts to re-establish the NetEx/eFT session during RNT processing. The accepted range of values is 1 to 32767 seconds.<br><br>The default is 1200 seconds.<br><br>The minimum spelling is -RNT_TIM |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-SPAce** (STRING) | This qualifier specifies the number of bytes of space to be allocated for a new data set.<br><br>Append a 'K' to the number to represent kilobytes or an 'M' to represent megabytes. For example, to allocate 2 megabytes of space for a data set, declare bytes as **-SPAce** 2M.<br><br>A **-SPAce** value MUST be declared unless the sending host internally provides an estimate of the file size to z/OS. Some systems such as OpenVMS, UNIX/Linux, or PC-DOS have this ability. If the source file is being sent from another z/OS host, the user must specify a **-SPAce** value since IBM z/OS NetEx/eFT does not internally provide the file size.<br><br>Preceding the bytes value with an asterisk (*) causes the bytes value to be used only if the sending host did not provide an estimate of the file size. Specifying a byte value without an asterisk always overrides the sending host's estimate.<br><br>The default is *16K.<br><br>The minimum spelling is -SPA. |
| **-STATistics** (BOOLEAN) | This qualifier causes NetEx/eFT to create ISPF statistics after writing a new or replaced MVS partitioned data set (PDS) member.<br><br>The default value is ON.<br><br>The minimum spelling is -STAT. |
| **-TAB** (BOOLEAN) | This qualifier causes tab characters (EBCDIC x'05' or ASCII x'09') to be expanded to the next tab expansion column. **-TAB** expansion is only active when **-MODe CHAracter** is specified. |
| **-UNIT** (STRING) | This qualifier tells NetEx/eFT to create the destination data set on a specific device, or a certain type or group of devices. The UNIT qualifier must be a 1- to 8-character alphanumeric string including national characters ($, #, or @ in the United States). If no unit name is specified, the unit name from the TSO/E user's profile is used.<br><br>When **-TAPE** is specified and **-VOLSER** is specified then **-CREate REPLACE** must be specified. Otherwise, when **-VOLSER** is not specified then **-CREate NEW**, **-CREate REPLACE**, or **-CREate DELETE** can be specified. |

| Table 1. z/OS File Transfer Qualifiers and Default Values | |
|---|---|
| **Qualifier** | **Description** |
| **-VOLUME** (STRING) | This qualifier tells NetEx/eFT to create the destination data set on a specific volume. The **VOLUME** qualifier must be a 1- to 6-character alphanumeric string including national characters ($, #, or @ in the United States). There is no default VOLUME name.<br><br>When **-TAPE** is specified and **-VOLSER** is specified then **-CREate REPLACE** must be specified. Otherwise, when **-VOLSER** is not specified then **-CREate NEW**, **-CREate REPLACE**, or **-CREate DELETE** can be specified. |
| **-WRAP** (BOOLEAN) | This qualifier indicates that data records received from the sending host that exceed the destination data set's record length should be wrapped into multiple records instead of being truncated. **-WRAP** is only used in CHARACTER mode transfers. If **-WRAP** is NO, long records will be truncated. An informative message is displayed if any records are wrapped or truncated.<br><br>The default is ON. |

# Definition of DIRECTORY Under IBM z/OS NetEx/eFT

When NetEx/eFT is invoked from either the Initiator or Responder side, the session begins with a default definition for both the **LOCAL** or **REMOTE** qualifier **TSOPREFIX** (or **DIRECTORY**). The definition of this qualifier is the user's default TSO/E prefix when logged onto the z/OS system. Any data set name accessed by NetEx/eFT gets prefixed with this value unless declared as a fully qualified data set name (enclosed in single quotes).

To change the default value of **TSOPREFIX** (or **DIRECTORY**), the NetEx/eFT commands **SET LOCAL TSOPREFIX** or **SET REMOTE TSOPREFIX** are used. For example, if the remote host is a z/OS system, the **TSOPREFIX** can be changed to EFTFILE, with the following command:

```
eFT> set remote tsoprefix EFTfile
```

To display the current value of TSOPREFIX, type:

```
eFT> show remote tsoprefix
eFT: TSOPREfix ............ EFTFILE
```

The **DIRECTORY** qualifier is equivalent to **TSOPREFIX** on z/OS hosts and is implemented on all NetEx/eFT hosts. All subsequent file transfers and remote commands will then use this new value as a default unless a fully qualified data set name is given.

**Note:** The **TSOPREFIX** is remembered across TSO/E sessions.

# IBM z/OS File or Data Set Name Specifications

A file on an IBM z/OS system is referred to as a "data set". NetEx/eFT will use these terms interchangeably.

Below is a very brief discussion of file or data set specification syntax supported by NetEx/eFT on IBM z/OS hosts. This is provided as an aid to the occasional z/OS user who is interested in transferring files to or from an IBM z/OS system yet is unsure of file specification syntax.

A data set specification under IBM z/OS has the form:

```
'PREFIX.QUAL1.QUAL2.QUAL3...QUALn(MEMBER)'/PASSWD
```

where,

**'...'** The single quotes are optional. You should only include them when explicitly providing a **PREFIX** (see below). A data set specification including a **PREFIX** and surrounded by single quotes is referred to as a "fully qualified data set name".

**PREFIX** (optional) is an alphanumeric string including national characters (such as $, #, and @ in the United States) from one to eight characters long with the first character being alphanumeric or a national character. If you explicitly provide a **PREFIX**, you must also put single quotes at the beginning and end of the data set name. If you leave off the single quotes, NetEx/eFT prefixes the data set name you specify with the value of the z/OS host's qualifier **TSOPREFIX**. **TSOPREFIX** is initially set to the prefix defined in the TSO/E user's profile.

**QUALx** is an alphanumeric string including national characters ($, #, or @ in the USA) from one to eight characters long, the first character being alphabetic or a national character. **QUALx** strings must be separated by a single period. All qualifiers are optional.

**(MEMBER)** (optional) represents the name of a member of a special kind of data set called a partitioned data set. A member name is an alphanumeric string including national characters ($, #, or @ in the USA) from one to eight characters long, the first character being an alphabetic or a national character. The member name must be enclosed in parentheses. When no member name is present, NetEx/eFT assumes the data set is a sequential data set.

**/PASSWD** (optional) optional data set password from one to eight alphanumeric or national characters long, the first of which must be alphabetic or a national character ($, #, or @ in the USA). The default password is the TSO/E user ID password.

The maximum length of a data set name (excluding optional member and single quotes) is forty-four characters.

Experienced IBM z/OS users may also use previously allocated data sets by specifying Data Definition (DD) Names instead of data set names. To specify a DD name, use the format:

```
DD:ddname
```

For example, to send the DD name "MJG001" from the command line you would type:

```
SEND DD:MJG001
```

The optional member name may be supplied following the DD name. Specifying a member name will temporarily override the member name provided when the DD name was originally allocated. The order of precedence for DD names is as follows:

1. A member name specified on the z/OS NetEx/eFT command, such as:

   ```
   DD:PDS(MEMBER)
   ```

2. A member name specified as part of the data set name on a TSO/E ALLOCATE command or JCL DD card, for example:

```
DSN=PREFIX.PDS(DDMBR)
```

3. A member name constructed from the file name on the remote host system. For example, a remote host file name of "DOCUMENT.TXT" would generate a member name of "(DOCUMENT)".

**Note:** Overriding the member name on a DD: specification prevents the data set from being extended beyond its current space allocation.

# IBM z/OS File or Data Set Specification Examples

Some examples of IBM z/OS data set specifications are listed below to help non-z/OS users get a better understanding of their appearance.

```
IBMNODE.FILE.EXT
'LANCE.FILES.SOURCE.FOR'
'LANCE.TEST.PROGRAMS.SAVE.FILE#A1.OLD'
FILE$001.#01
TEST.PROGRAMS(MEMBER01)
'LANCE.PDS.DEVELOP(LOOPA)'
'LANCE.PDS.FILES(LOOPB)'/G$ZIP@
```

## File Transfer Examples from a Local IBM z/OS Host

### Example 1:

To receive file "ABC.FOR" from the remote host into a sequential data set on the local IBM z/OS system, issue the following:

```
eFT> receive abc.for
eFT: SOURCE               DESTINATION           SIZE
eFT: ------------------------------------------------
eFT: ABC.FOR              SMITHX.ABC.FOR       32100
```

Since no destination file or data set name was specified, the data set was given the same name as the source file name and created with the current value of **TSOPREFIX** (e.g., "SMITHX") prepended to it.

### Example 2:

To send the member "ABC" of the partitioned data set "EFT.SRC.FOR" to a remote host the following send command is issued:

```
eFT> send 'EFT.src.for(abc)'
eFT: SOURCE                   DESTINATION      SIZE
eFT: ------------------------------------------------
eFT: EFT.SRC.FOR(ABC)         ABC.FOR         32100
```

### Example 3:

To send the sequential data set "DOC.TXT" using **BACKUP** mode to preserve the length and contents of the data set, the following command is issued:

```
eFT> send -mode backup doc.txt
eFT: SOURCE                  DESTINATION         SIZE
eFT: ------------------------------------------------
eFT: SMITHX.DOC.TXT          DOC.TXT            56700
```

**RESTORE** mode can now be used to restore the file "DOC.TXT" in its original form. Notice that the default **TSOPREFIX** (e.g., "SMITHX") prefixes the source file name "DOC.TXT".

## File Transfer Examples to a Remote IBM z/OS Host

### Example 1:

To receive the file "ABC.FOR" from the remote host as described above into an existing partitioned data set named "EFT.SRC.FOR", the following command is issued:

```
eFT> receive -create replace abc.for 'EFT.src.for'
eFT: SOURCE              DESTINATION           SIZE
eFT: ------------------------------------------------
eFT: ABC.FOR             EFT.SRC.FOR(ABC)     32100
```

Notice that the **TSOPREFIX** was not added to the destination data set name since it was enclosed in single quotes. Also, since "EFT.SRC.FOR" was a partitioned data set and no member name was given, NetEx/eFT used the file name portion of "ABC.FOR" (which is "ABC") as the member name to create in the data set. The **-CREate REPLACE** option tells NetEx/eFT to replace an existing member, should one exist.

### Example 2:

To send a local file called "MYFILE.TMP" from a non-z/OS system to an IBM z/OS host, forcing **CRC** on the transfer, the following command is issued:

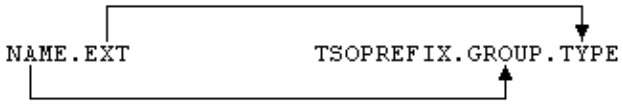
```
eFT> send myfile.tmp -crc
eFT: SOURCE              DESTINATION             SIZE
eFT: ------------------------------------------------
eFT: MYFILE.TMP          SMITHX.MYFILE.TMP     232100
```

Since no destination file or data set name was specified, the data set was given the same name as the source file name and created with the current value of **TSOPREFIX** (e.g., "SMITHX") prepended to it.

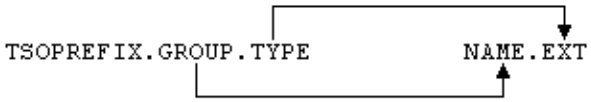
# Converting from File Names to Data Set Names

To convert from standard file name syntax "FILE.EXT" (FILENAME followed by EXTENSION, as used by OpenVMS, UNIX, PC-DOS, IBM VM, etc.) to z/OS data set names, and back again, NetEx/eFT makes some assumptions about data set naming conventions. In order to satisfy a wide range of z/OS users, NetEx/eFT tries to conform to the ISPF library convention which names individual qualifiers as "PROJECT.GROUP.TYPE" followed by an optional member name for partitioned data sets. The conversion then is straight forward. That is, the file EXTENSION maps to data set TYPE. File NAME maps to the GROUP name for sequential data sets or to MEMBER name for partitioned data sets. Pictorially, the mapping is:

Conversion to/from a Sequential Data Set Name:

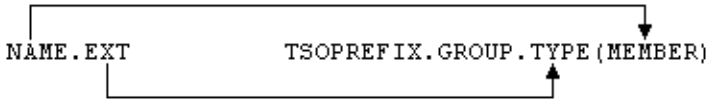



and,

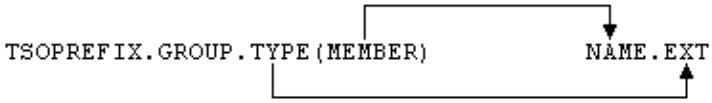



For example, if a user were to send the file "PROGRAM.FOR" to a z/OS host with a **TSOPREFIX** of "SMITH", the resulting data set name would be "SMITH.PROGRAM.FOR".

Conversion to/from a Partitioned Data Set Name:




and,




For example, if a user were to send data set member "SMITH.FOR(PROGRAM)" from a z/OS host, the resulting file would be "PROGRAM.FOR".

This conversion takes place at file transfer time when no destination file is specified on the **SEND**/**RECEIVE** command line. It assumes TYPE is always the last element or qualifier of a data set name and that it can be mapped to an EXTENSION. If these conversion rules do not map appropriately for a data set, it is always safe to specify a fully qualified data set name (or entire path and file name) as the destination file specification.

Note also above that the current value of **TSOPREFIX** (which is equivalent to **DIRECTORY**) is prefixed onto any destination data set name that is built unless it is a fully qualified data set (enclosed in apostrophes).

See a description of the "SENd Command" and "RECeive Command" for IBM z/OS for further discussion on this topic.

# Source Wildcard Support for IBM z/OS File Transfers

Wildcarding is valid on the source file specification for both the **SEND** and **RECEIVE** commands. Two NetEx/eFT wildcard characters have been defined in an attempt to standardize the wildcarding for all hosts which can support it. These are:

***** matches zero or more characters within the current data set name level or member name of a partitioned data set. For example, "XXX.*DEF" matches the data sets "XXX.ABCDEF,XXX.CDEF", and "XXX.DEF", but not "XXX.DEF.ABC".

****** Two asterisks together matches zero or more characters in all levels of a data set name. For example, "XXX.XYZ**" matches the data sets "XXX.XYZ123", "XXX.XYZ123.ABC.DEF", "XXX.XYZ.ABC", and "XXX.XYZ".

**?** matches exactly one character within the current data set name level or member name of a partitioned data set. For example, "XXX.?DEF" matches the data sets "XXX.ADEF", "XXX.BDEF", "XXX.CDEF", but does not match "XXX.ABCDEF" or "XXX.DEF".

IBM z/OS NetEx/eFT supports wildcarding within data set names and member names of partitioned data sets. The wildcard characters can be combined to search multiple data set name levels. Wildcarding is not allowed in the first level of a fully qualified data set name.

Wildcarding under NetEx/eFT for IBM z/OS can be used to match sequential data set specifications, partitioned data set specifications, or both. NetEx/eFT matches on only sequential data sets when no member specification exists. To tell NetEx/eFT to match only on partitioned data sets (and their members), append a member specification at the end of the data set qualifiers. Within the parenthesis must be either additional wildcard characters or a valid member name. By default, NetEx/eFT looks for, and matches only on, sequential data sets or partitioned data sets, but not both. To instruct NetEx/eFT to match on both, a special syntax has been introduced. Instead of using parenthesis around a member name, use angle brackets ('<' and '>'). Any data set that matches the qualified data set name is considered a match; if the data set is a partitioned data set the additional specification between the angle brackets is used to match any member names within a matching partitioned data set.

An example use of source wildcarding for IBM z/OS NetEx/eFT appears below. The example demonstrates sending all the sequential data sets with the default TSO/E prefix that contain the character sequence "ONE" somewhere in their name:

```
eFT> send **ONE**
eFT: SOURCE                    DESTINATION            SIZE
eFT: ------------------------------------  ----------
eFT: SMITHX.CHAP.ONEX          CHAP.ONEX          253991
eFT: SMITHX.AAONEAA            AAONEAA             48228
eFT: SMITHX.ONE.TWO.THREE      TWO.THREE          774263
eFT: SMITHX.A.B.C.BONE.D       BONE.D                776
```

The following example demonstrates the use of wildcarding within a data set name and member name of a partitioned data set. In this example, a fully qualified data set name is specified so the first level cannot contain wildcard characters:

```
eFT> send 'JIM.*A*(*DO?)'
eFT: SOURCE                    DESTINATION            SIZE
eFT: ------------------------------------------------------
eFT: JIM.APPLE(MYDOW)          MYDOW.APPLE         84732
eFT: JIM.UA(TESTDOC)           TESTDOC.UA          32768
eFT: JIM.A(TESTDOC)            TESTDOC.A            5549
eFT: JIM.MACRO(DOA)            DOA.MACRO          111488
```

The source specification "JIM.*A*(*DO?)" contains a member name between parenthesis, therefore only partitioned data sets match the given source specification. When a partitioned data set is found that matches, the appropriate members of that data set are transferred depending upon the member name. The member specification may contain both an asterisk ('*') and a question mark ('?') for wildcarding on members within a partitioned data set.

IBM z/OS wildcarding also permits wildcarding on both sequential and partitioned data sets at the same time by introducing a new syntax for IBM z/OS data set specifications. If the data set specification contains a member name between angle brackets ('<' and '>'), then the specification matches both sequential and partitioned data sets and only uses the member name if the data set is a partitioned data set. For example, the following SEND command transfers both sequential data sets and members from a partitioned data set that match the wildcard source specification:

```
eFT> send 'PRODUCT.*.DOC<*>'
eFT: SOURCE                        DESTINATION          SIZE
eFT: ------------------------------------------------------
eFT: PRODUCT.INSTALL.DOC           INSTALL.DOC           4732
eFT: PRODUCT.MEMO.DOC              MEMO.DOC             32768
eFT: PRODUCT.MANUAL.DOC(CHP1)      CHP1.DOC              5549
eFT: PRODUCT.MANUAL.DOC(CHP2)      CHP2.DOC             11488
eFT: PRODUCT.MANUAL.DOC(INTRO)     INTRO.DOC             5345
eFT: PRODUCT.MANUAL.DOC(SUM)       SUM.DOC               4232
```

# Destination Wildcard Support for IBM z/OS File Transfers

Destination wildcarding is also available on IBM z/OS NetEx/eFT. Destination wildcarding makes it possible to transfer a set of files from one system to a z/OS host, modifying the file or data set names as part of the process. The single character '*' is used to make this happen.

When an '*' is seen as part of the destination file specification, NetEx/eFT replaces it with either the "file name" portion of the source file specification or the "file extension" portion of the source file specification, depending on its position in the destination file specification. For example, to send all files with an extension of FTN from some local host to a z/OS host, renaming the files with an extension of FOR, destination wildcarding would be used as:

```
eFT>.send *.ftn *.for
eFT: SOURCE          DESTINATION                    SIZE
eFT: ------------------------------------------------------
eFT: ABCDEF.FTN      GUEST.ABCDEF.FOR             984744
eFT: SAMPLE.FTN      GUEST.SAMPLE.FOR             668264
eFT: TEST.FTN        GUEST.TEST.FOR               120288
```

In this example, NetEx/eFT replaced the "*" in the destination file specification with file names ("ABCDEF", "SAMPLE", and "TEST") from the source file specifications. The last qualifier name was also renamed from "FTN" to "FOR".

It is also possible to append characters around the "*" as well as use destination wildcarding to a partitioned data set. For instance, the destination file specification could have appeared as:

```
eFT> send *.ftn x*x(y*)
eFT: SOURCE            DESTINATION                  SIZE
eFT: ------------------------------------------------------
eFT: ABCDEF.FTN        GUEST.XFTNX(YABCDEF)       983744
eFT: SAMPLE.FTN        GUEST.XFTNX(YSAMPLE)       668264
eFT: TEST.FTN          GUEST.XFTNX(YTEST)         120288
```

In this example, destination wildcarding was used to modify both the file name and file extension portion of the destination file.

# Transfer Modes Supported Under IBM z/OS NetEx/eFT

IBM z/OS NetEx/eFT supports seven modes of file transfer: **BACKUP**, **CHARACTER**, **COPY**, **RECORD**, **RESTORE**, **STREAM**, and **V1CHAR**. A user selects the file transfer mode by setting the **SEND** or **RECEIVE** qualifier **-MODE**. A mode must be supported by both hosts for a successful file transfer. The **-MODE** qualifier defines the form in which data will be transferred. Keep in mind that the internal representation of data within a file varies from host to host even though most hosts define the same modes of transfer. Each mode is described in further detail below as it relates to IBM z/OS.

| Table 2. Supported z/OS Transfer Modes | |
|---|---|
| **Mode** | **Description** |
| **BACKUP** | This mode is designed to allow IBM z/OS files or data sets to be backed up on some host regardless of word size, then restored at a later time. A special header is built around the resulting file in order to properly restore the file. The resulting file will generally not be usable on the destination host but can be restored to a z/OS system with its original data and size. **BACKUP** mode under IBM z/OS NetEx/eFT does not save data set characteristics as part of the backup header. It simply ensures that all data backed up can be restored without loss of data or with extra data due to word size incompatibilities between hosts. |
| **CHARACTER** | This mode is generally designed for moving text files from one host to another. This mode performs automatic code conversion across the network and assumes the data being transferred contains only text data. An error will usually result if an attempt to transfer binary data is made. The qualifier **-MAXRECord** determines the maximum allowable record that can be read from or written to a data set. Fixed length records are padded with spaces when written to a data set and stripped of trailing spaces before being sent across the network. |
| **COPY** | This mode is designed for peer-to-peer file transfers. In **COPY** mode, IBM z/OS files can be moved from one z/OS system to another. Most types of file (text or binary) can be transferred very fast in **COPY** mode since file access is done as efficiently as possible, without individual records having to be manipulated. The data in **COPY** mode is transferred as an unstructured stream of bytes. |
| **RECORD** | This mode is for moving IBM z/OS record-oriented binary data. No code conversion is performed on **RECORD** mode transfers. This mode would typically be used anytime a data set containing non-text data was to be transferred across the network. It is recommended that data sets transferred in **RECORD** mode have variable length records as opposed to fixed length. Fixed length records are padded with nulls when written, but not stripped when read back. |
| **RESTORE** | This mode is used to restore a file previously transferred in **BACKUP** mode. **RESTORE** mode expects to find a backup header built around the file it is attempting to restore. Since **BACKUP** mode under IBM z/OS NetEx/eFT does not save data set characteristics, a backed-up data set gets restored using the **SEND**/**RECEIVE** qualifier values at the time of the transfer. For instance, if a fixed blocked data set is saved using **BACKUP** mode, the same data set should be restored by setting the **-RECFORmat** qualifier to FB. Refer to the discussion on **BACKUP** mode above. |

| Table 2. Supported z/OS Transfer Modes | |
|---|---|
| **Mode** | **Description** |
| **STREAM** | This mode supports any z/OS data set organization and format. Stream data is treated as a literal stream of data, without regard to record separators or lengths. Creating an MVS stream data set will create records of the maximum allowable length. For fixed length records this will be the local record length (LRECL), for variable length records this will be LRECL-4, and for undefined length records this will be the block size (BLKSIZE). **STREAM** mode data sets can be used to receive binary data on an MVS system, without having MVS record formats affect the remote system's data. Most native MVS data sets transferred in stream mode will appear to become "packed" when transferred to another host. The only meaningful use of stream mode data in an MVS to MVS transfer is when using fixed length records with the same LRECL on both MVS hosts. |
| **V1CHAR** | This mode is provided for compatibility with Release 1 versions of NetEx/eFT. It is equivalent to **CHARACTER** mode. **CHARACTER** mode under Release 1 used a different internal format. |

# Creating Data Sets with NetEx/eFT

Data sets can be created with NetEx/eFT in two ways. First, a user may create a data set on a z/OS host using the **LOCAL** or **REMOTE** TSO/E ALLOCATE command. Typically, one would do this to define additional data set characteristics not available through NetEx/eFT. Once a data set is created, it can be referenced through NetEx/eFT by data set name or allocated DD name. To transfer a file into a previously allocated data set, specify the **SEND**/**RECEIVE** qualifier **CREATE** with the value **REPLACE** (overwrite existing data set).

The second and easiest way to create a data set with NetEx/eFT is to simply transfer a file (using **SEND** or **RECEIVE**) into a nonexistent data set. The size of this newly created data set will be based on the value of the SPACE qualifier provided by the user. If this value is zero, or begins with an asterisk (*), NetEx/eFT will attempt to determine the size of the original file and allocate the data set using that size as a guide. See the **SEND**/**RECEIVE SPACE** qualifier for more information on specifying the size of a newly created data set in "Command Descriptions" on page 117.

NetEx/eFT uses the current values of the qualifiers **RECFORMAT**, **BLOCKSIZE**, and **RECLENGTH** when creating a new data set via **SEND** or **RECEIVE**. These values must be compatible for NetEx/eFT to allocate space and create the data set. To be compatible, NetEx/eFT makes the following checks:

```
RECFORMAT 'F' (fixed):
  BLOCKSIZE must equal RECLENGTH

RECFORMAT 'FB' (fixed blocked):
  BLOCKSIZE must be a multiple of RECLENGTH

RECFORMAT 'V' or 'VB' (variable or variable blocked):
  BLOCKSIZE must be greater than or equal to (RECLENGTH + 8)

RECFORMAT 'VBS' (variable blocked spanned):
  RECLENGTH may exceed BLOCKSIZE

RECFORMAT 'U' (undefined):
  RECLENGTH is not checked against BLOCKSIZE
```

Regular z/OS users should note that for variable length records **RECLENGTH** represents the length of user data not including record or block headers. That is, **RECLENGTH** is really (LRECL-4).

These same validations are also done on existing data sets that are to be read or overwritten by NetEx/eFT.

# IBM z/OS Data Set Organizations Supported by NetEx/eFT

NetEx/eFT supports two types of z/OS data set organizations, Sequential and Partitioned. Entire sequential data sets may be transferred from a z/OS host to any other host on the network. Partitioned data sets on the other hand cannot be transferred as a single file. A member name must be specified. However, using wild-carding, an entire partitioned data set may be transferred member-by-member. Partitioned data set directory information is not processed by NetEx/eFT for IBM z/OS and will not be transferred when a member is sent.

There is currently no support in z/OS NetEx/eFT for transferring concatenated data sets or VSAM data sets in native format. VSAM data sets can be unloaded to a sequential format (IDCAMS Export), transferred to another host, and reloaded (IDCAMS Import).

# IBM z/OS Record Formats Supported by NetEx/eFT

NetEx/eFT supports data sets of Fixed, Variable and Undefined length records. In addition, these data sets may have record characteristics such as Spanned, Blocked, ANSI or Machine code control characters. When creating a new data set, you declare its record format (RECFM) by setting the **SEND** or **RECEIVE** qualifier **RECFORMAT**. This value must also be compatible with the qualifier values for **BLOCKSIZE** and **RECLENGTH**. See **SEND**/**RECEIVE** qualifiers for more information on specifying record formats to NetEx/eFT in “File Handling Under IBM z/OS NetEx/eFT” on page 41.

# Advanced Local User's Guide

## Introduction

This section is intended for users who already have a good working knowledge of NetEx/eFT and would like to learn more details about the product. Site administrators responsible for NetEx/eFT as well as those users developing NetEx/eFT scripts and aliases will benefit most from this section.

Most of this section discusses how to develop a custom NetEx/eFT interface using string functions, input scripts, and aliases. The remainder of the section discusses advanced topics such as user-definable help files and NetEx/eFT batch jobs.

## Special Characters

Several characters have special meaning to NetEx/eFT when it is parsing a command line. The position of the character within a line is a determining factor on how NetEx/eFT will interpret it. The characters are:

***** The asterisk is treated as a comment character if it appears as the first character on the command line. That is, NetEx/eFT ignores the line. An alternate comment character is the pound sign character ('#' in the United States). Comments are generally used within NetEx/eFT alias definitions and input files to make them more readable for the user. The following are example NetEx/eFT comment lines:

```
eFT> * This is a comment and is ignored
eFT> # The pound sign is treated as a comment too
```

**#** The pound sign is identical to the '*' character as described above.

**-** The dash character has two meanings within NetEx/eFT. First, if it appears as the last character of a command line, it tells NetEx/eFT to continue the command on the next line. NetEx/eFT then prompts for more input. For example:

```
eFT> set alias example -
More>> text Example of continuing a command on next line.
```

The second use of the dash character is to specify a qualifier to a NetEx/eFT command. A qualifier must follow the dash without any spaces between the two. For example, to turn on quiet mode on the **SEND** command, the user would specify the **QUIET** qualifier as below:

```
eFT> send -quiet source_file destination_file
```

To tell NetEx/eFT to take the dash literally on a command line, escape it by typing two dashes in a row (i.e., '--').

**!** The exclamation point is used by NetEx/eFT as the escape character for special command line processing. Depending on its position within a command line, it is interpreted several different ways.

First, the exclamation point is used to create multi-command aliases when it appears as the last character on the command line (with no trailing spaces). For example, to create a two-command alias called NAME, the exclamation point is used as follows:

```
eFT> set alias NAME {} ask -prompt "Enter Name: " name !
More>> text Hello {name}.
```

Second, the exclamation point is used to escape the '{' and '}' characters. An exclamation point appearing immediately before either of these characters tells NetEx/eFT to take them literally and skip any string processing that would normally be done. For example:

```
eFT> text Leave the braces !{here!}.
```

Finally, the exclamation point is used to tell NetEx/eFT not to do any alias processing on a given command. Since an alias may have the same name as a NetEx/eFT command, an exclamation point immediately preceding a command tells NetEx/eFT to use the command, not the alias. The same holds true for local and remote command aliases. If an exclamation point appears immediately before a command preceded by **LOCAL** or **REMOTE**, NetEx/eFT uses the command as it appears without processing it as an alias. Each of the following lines in the example below tells NetEx/eFT to use the command even if an alias by the same name has been defined:

```
eFT> !text ignore alias processing on text
eFT> local !dir
eFT> remote !who
```

**{** This character marks the beginning of string substitution. It is used along with the '}' character to delimit a positional parameter, a string variable, or a string function. For example, to print out the value of string variable "NAME", the following command could be issued:

```
eFT> text Your name is {name}.
```

To tell NetEx/eFT to take either the '{' or '}' literally, use the exclamation point:

```
eFT> text Print the line with braces !{name!}.
```

To tell NetEx/eFT to turn off string substitution, the sequence '{}' is used:

```
eFT> text {} Turn off string substitution {name}.
```

**}** This character marks the end of string substitution. See the explanation of '{' above.

**"** The double quote character allows the user to create a string that contains embedded blanks:

```
eFT> set input prompt "NEW PROMPT> "
```

To escape the double quote character, type two in a row (" ")

# NetEx/eFT String Substitution

NetEx/eFT string substitution gives users the ability to write complex aliases and input scripts. String substitution can take place anywhere within a NetEx/eFT command line. The syntax is:

```
{string}
```

where "string" is either a string literal, string variable (including positional parameters), or a string function. String substitution involves the replacement of `{string}` by its computed value. The result, or replaced value, of string substitution is always a string.

A string literal refers to any quoted string. The following are examples of a string literal:

```
" "
"Box"
"Big Box"
"This is a Big Box"
```

Performing string substitution on these string literals within a NetEx/eFT **TEXT** command produces the following results:

```
eFT> text {" "}.
eFT:  .

eFT> text {"Box"}.
eFT: Box.

eFT> text {" Big Box "}.
eFT:  Big Box .

eFT> text {"This is a Big Box"}.
eFT: This is a Big Box.
```

A string variable, also referred to simply as a variable, is an arbitrary name that is associated with a predefined character string value. Assume the following string variables exist and are defined as indicated:

| Variable | Definition |
|---|---|
| Hostname | BLUESKY |
| A | Sample string |
| Day | 28 |

String substitution involves the replacement of a string variable by its currently assigned value. Therefore, performing string substitution on these variables within the **TEXT** command, produces the following results:

```
eFT> text {hostname}.
eFT: BLUESKY.

eFT> text {a}.
eFT: Sample string.

eFT> text {day}.
eFT: 28.
```

A string function refers to one of the NetEx/eFT defined functions that may accept a string as parameter and return a string as a result. A few simple string functions with sample arguments appear below:

```
date()
upper("this is a test")
cmp("good", "bad", "Compared", "Didn't compare")
```

Performing string substitution on these example string functions result in the following:

```
eFT> text {date()}.
eFT: Sun Thu Jun  6, 2019.

eFT> text {upper("this is a test")}.
eFT: THIS IS A TEST.

eFT> text {cmp("good", "bad", "Compared", "Didn't compare")}.
eFT: Didn't compare.
```

Although the **TEXT** command was used in the examples above, string substitution can be performed anywhere within a NetEx/eFT command line, whether it is part of another NetEx/eFT command, or on a line by itself. It's important to remember that the result of any string substitution is simply another string. Therefore, the resulting string could even be a NetEx/eFT command.

## String Variables

NetEx/eFT variable names can be from one to twenty alphanumeric characters long, including underscores and similar special characters. There are two types of variables, local and global. A local variable exists only within the input level, or input file, in which it was initially defined. If an input file is nested, it cannot reference local variables defined by its caller. A local variable defined within an input file is no longer valid after that input file is exited.

A global variable can be defined from any input level, or input file, and referenced by any other one. That is, once a global variable is defined within a NetEx/eFT session, that variable is known throughout the session, regardless of the current input level. Generally, it is better to use local variables whenever possible since these do not get left around from input file to input file. Global variables, on the other hand, take up NetEx/eFT internal storage and can eventually lead to an "Environment overflow" condition. This condition may be relieved by undefining some previously defined global variables as described later in this section. This will recover internal storage space, even though the undefined variable will still be displayed with a null value.

Variables can be defined in a couple of ways. The most obvious is with the **SET VARIABLE** and **SET GLOBAL** commands. **SET VARIABLE** is used to define a local variable. **SET GLOBAL** defines a global variable. An example of each of these appears below:

```
eFT> set variable username smith
```

and,

```
eFT> set global days Saturday and Sunday
```

In the first case, local variable "username" was given the value "smith". In the second case global variable "days" was assigned the value "Saturday and Sunday". Keep in mind, all variables are defined as character strings. To show the current value of the variables defined above use the **SHOW VARIABLE** and **SHOW GLOBAL** commands respectively:

```
eFT> show variable username
eFT: USERNAME .......... smith
```

and,

```
eFT> show global days
eFT: DAYS .............. Saturday and Sunday
```

To undefine a local or global variable, use the SET command with the variable name and no value. For example, to undefine the two variables described above, use the following commands:

```
eFT> set variable username
```

and,

```
eFT> set global days
```

An undefined variable will appear in a **SHOW VARIABLE** or **SHOW GLOBAL** display as a variable without a definition. If an undefined variable is referenced within a NetEx/eFT command, a null string is substituted in its place.

The second way to define a local variable is with the **ASK** command. The example below defines the variable "username" again, but using **ASK**:

```
eFT> ask -prompt "Enter Username: " username
Enter Username: smith
```

The real significance of string variables is the ability to use them within NetEx/eFT aliases and input scripts. To reference the value of a variable, enclose the variable name in braces "{}" within a NetEx/eFT command

line (this invokes string substitution). Refer back to the variables "username" and "days" above. Their values can be used in the **TEXT** command as:

```
eFT> text The value of variable username is {username}.
eFT: The value of variable username is smith.
```

and,

```
eFT> text {days} are coming soon.
eFT: Saturday and Sunday are coming soon.
```

The braces around the variable name tell NetEx/eFT to replace it with its assigned value.

Since local and global variables are stored differently within NetEx/eFT, it is possible to create a global variable with the same name as a local variable. For example:

```
eFT> set variable hostname alpha
eFT> set global hostname omega
```

The variable "hostname" has been defined twice, once as a local variable with a value of "alpha" and again as a global variable with a value of "omega". Because NetEx/eFT gives precedence to local variables, referencing "{hostname}" will result in the local value of "alpha". A special syntax is used to reference the value of a global variable when a local variable of the same name exists. An example follows:

```
eFT> text The local value is {hostname}.
eFT: The local value is alpha.
eFT> text The global value is {hostname:global}.
eFT: The global value is omega.
```

By default, when a variable is enclosed in braces (without the **:global** syntax), NetEx/eFT looks for a local variable by that name. If one is found, its value is returned. If one is not found, NetEx/eFT looks next for a global variable by the same name and uses its value if found. Appending the variable name with **:global** within the braces tells NetEx/eFT to look only for a global variable of that name.

NetEx/eFT carries this special syntax one step further in allowing the substitution of NetEx/eFT command qualifier values. These values can be used as variables as shown below. The syntax is:

```
{qualifier:cmd}
```

where **qualifier** is a valid qualifier (including informational qualifiers) for the specified command **cmd**. For example, if the current value of the **SEND** qualifier **CREATE** was defined to be "new", this could be referenced as:

```
eFT> text SEND qualifier CREATE is {create:send}.
eFT: SEND qualifier CREATE is new.
```

The following example shows how the command qualifiers can be used to set the NetEx/eFT input prompt. Since the prompt is controlled by the **INPUT** command qualifier **PROMPT**, this can be modified to the user's liking. To change the prompt from the default of "`eFT>` " to the current name of the remote host (assume it's called "STARMAN"), the following command is used:

```
eFT> set input prompt {} {host:remote}>
STARMAN>
```

The syntax **{host:remote}** says to extract the value of informational qualifier **HOST** from the **REMOTE** command defaults, and replace this value on the command line. (The empty "{}" is explained in the section entitled "Disabling String Substitution" on page 91.) The following command produces an equivalent result:

```
eFT> set input prompt STARMAN>
STARMAN>
```

Although the result is equivalent, the second example above does not allow for flexibility within an alias or input script, nor is it flexible enough to change for each connection made to a different remote host.

## String Literals

A string literal as mentioned earlier, is any quoted string. Quoted strings refer to a string of characters, enclosed in double quotes, from zero to *n* long, where *n* is arbitrarily long depending on the space remaining in the NetEx/eFT input buffer.

Below is an example of a string literal used with string substitution and the string function **LOWER()** (described on page 85) to define the NetEx/eFT prompt as " command? " with two leading and two trailing spaces:

```
eFT> set input prompt {lower("  COMMAND?  ")}
```

This will result in the following prompt, including the two leading and two trailing spaces:

```
  command?
```

In order to have embedded double quotes within a string literal, the user must escape each one with a second double quote. The example below shows this:

```
eFT> set input prompt {lower("""Enter a Command:"" ")}
```

The resulting prompt would be:

```
"enter a command: "
```

with a single trailing space.

The examples above demonstrate the use of string literals within the **SET** command. Quoted strings within a **TEXT** command, however, are taken literally. Therefore, to display the same "enter a command:" with double quotes using the **TEXT** command, the following syntax is used:

```
eFT> text "enter a command:"
eFT: "enter a command:"
```

## String Functions

NetEx/eFT string functions perform certain predefined tasks and return a string as output. String functions perform such tasks as comparing two strings, forcing a string to upper/lower case, returning the status of the previous command, and sleeping for a predetermined amount of time. Some string functions require arguments, and some do not. All arguments passed to a string function must be a numeric constant, a string literal, a string variable, or another string function. For example, the string function **lower()** takes a single argument which is a string that will be forced to all lower case characters. The following are all valid arguments:

```
lower("sample string")   - a string literal
lower(hostname)          - a variable named hostname
lower(date())            - a string function date()
lower(ext(time(),1,5))   - string functions with numeric constants
```

NetEx/eFT performs string substitution from the inside out. Therefore, if a string function exists as an argument to another string function, the innermost string function is executed first, and the resultant string is passed as an argument to the outer string function. In the example "lower(date())" above, the **date()** function would get processed first, then the actual date string would be passed as an argument to **lower()**.

The greatest use of string functions comes within NetEx/eFT scripts (input files or aliases). Often it is desirable to perform a NetEx/eFT command based on a certain condition. String functions make this possible. The following is an example of a simple script that tests the results of a command with the **status()** function and operates accordingly:

```
set input continue on
*
*     Loop until successful connection.
*
again:
ask -prompt "Hostname? " host
connect -quiet {host}
{eqs(status(), "S", "text Connect worked.", "goto again")}
```

This script also makes use of the **eqs()** function. **Eqs()** compares the result of **status()** with the string "S" (Success). If the strings compare (i.e., if the **CONNECT** was successful), the third argument of **eqs()** is used to replace the function in the substitution. If the strings do not compare, the fourth argument is used.

Notice that these last two arguments are simply NetEx/eFT commands enclosed in double quotes. The third argument "text Connect worked." prints a message at the user's terminal and continues processing. The fourth argument "goto again", causes processing to loop back to the "again:" label where the user is prompted for a new hostname. The **GOTO** command is discussed in a later section.

The remainder of this section describes the NetEx/eFT string functions in more detail. The functions are listed in alphabetical order except where functions are grouped by a logical association (for example, arithmetic operations).

The descriptions assume the user is familiar with NetEx/eFT strings and string variables as described in the section entitled "String Variables" on page 66. What follows is a list of the available string functions. Most of these functions follow the table in alphabetical order, however the arithmetic and logical functions are grouped together.

**Table 3. List of String Functions**

| Function | Description | Page |
|---|---|---|
| ADD | Adds two numeric string expressions and returns the result. | 72 |
| CHR | Returns a single character represented by the specified number in the local host machine's native character set (ASCII or EBCDIC). | 73 |
| CMP | Compares two strings. Allows for partial string match by specifying the required characters in upper case. | 74 |
| DATE | Returns the system date of the local host. | 75 |
| DEC | Subtracts one from a numeric string expression and returns the result. | 76 |
| DFN | Tests if a variable is defined. | 77 |
| DIV | Divides the first numeric string expression specified by the second and returns the result. | 72 |
| ENCrypt | Used to encrypt user passwords which will be used by NetEx/eFT to establish connections to remote hosts. | 78 |

| Table 3. List of String Functions | | |
|---|---|---|
| **Function** | **Description** | **Page** |
| ENV | Returns the value of the local host environment variable if the local host sup-ports such variables. If local host environment variables are not sup-ported or if the specified variable is not defined, a null string is returned. | 79 |
| EQ | Tests if the first number specified is equal to the second number specified. | 84 |
| EQS | Tests if the first string specified is equal to the second string specified. | 80 |
| EXT | Extracts and returns a bounded sequence of characters from a string. | 81 |
| GE | Tests if the first number specified is greater than or equal to the second. | 84 |
| GT | Tests if the first number specified is greater than or equal to the second. | 84 |
| INC | Tests if the first number specified is greater than the second. | 76 |
| INDEX | Returns the position of the second string specified within the first string. The function returns zero if the second string is not found. | 82 |
| LE | Tests if the first number specified is less than or equal to the second. | 84 |
| LEN | Returns the count of characters that make up the specified string. | 83 |
| LOWER | Returns the lower-case equivalent of a string expression, all characters but upper case are left untouched. | 85 |
| LT | Tests if the first number specified is less than the second. | 84 |
| MOD | Returns the remainder of the division of the first numeric string expression specified by the second. | 72 |
| MUL | Multiplies two numeric string expressions and returns the result. | 72 |
| MSG | Returns the information of the last message. MSG is most often used to tailor the NetEx/eFT output message format | 86 |
| NDF | Tests if a variable is not defined. | 77 |
| NE | Tests if the first number specified is not equal to the second. | 84 |
| NES | Tests if the first string specified is not equal to the second. | 80 |
| PARAMS | Substitutes the positional parameters specified. It is important to note that quoted parameters remain quoted and are considered one whole parameter – regardless of imbedded spaces. | 87 |

| Table 3. List of String Functions | | |
|---|---|---|
| **Function** | **Description** | **Page** |
| SLEEP | Causes NetEx/eFT to pause or "sleep" for a specified number of seconds. This process is not interruptible. This function results in a null string | 88 |
| STATUS | Returns the single status character of the previous command:<br><br>S = Success, E = Error. | 89 |
| SUB | Subtracts the second numeric string expression specified from the first and returns the result. | 72 |
| TIME | Returns the system time of the local host. | 90 |
| UPPER | Returns the upper-case equivalent of a string expression, all characters but lower case are left untouched. | 85 |

## Arithmetic Operations

The following is a list of arithmetic operators.

**ADD** adds two numeric string expressions and return the result.
**DIV** divide the first numeric string expression specified by the second and return the result.
**MOD** return the remainder of the division of the first numeric string expression specified by the second.
**MUL** multiply two numeric string expressions and return the result.
**SUB** subtract the second numeric string expression specified from the first and return the result.

**Format:**

```
add(number1, number2)
div(number1, number2)
mod(number1, number2)
mul(number1, number2)
sub(number1, number2)
```

Where:

**number1, number2** numbers to be operated on.

**Examples:**

Add two constants and return the result:

```
eFT> text {add(5,10)}
eFT: 15
```

Ask the user to enter three numbers:

```
eFT> ask -prompt "Enter 3 #'s: " num1 num2 num3
eFT: Enter 3 #'s: :usr.2 3 4
```

Find the square of the first number:

```
eFT> text The Square of {num1} is: {mul(num1,num1)
eFT: The Square of 2 is: 4
```

Find the total sum of the three numbers:

```
eFT> text {num1}+{num2}+{num3} {add(num1,add(num2,num3))}
eFT: 2+3+4 = 9
```

Divide the third number by the first number:

```
eFT> text {num3}/{num1} = {div(num3,num1)}
eFT: 4/2 = 2
```

## CHR Function

The CHR function returns a single character represented by the specified number in the local host's native character set (ASCII or EBCDIC).

**Format:**

```
chr(number)
```

Where:

**number** a number corresponding to the host's native character set.

**Examples:**

To display the quote character on a system on an ASCII host:

```
eFT> text This is a quote {chr(34)}
eFT: This is a quote "
```

To display the quote character on a system on an EBCDIC host:

```
eFT> text This is a quote {chr(127)}
eFT: This is a quote "
eFT> text This is a quote {chr(0x7F)}
eFT: This is a quote "
```

## CMP Function

The CMP function compares two strings. It allows for partial string match by specifying the required characters in upper case.

**Format:**

```
cmp(string, key, if_true [, if_false])
```

Where:

**string** a string expression whose letters are compared with the argument key.

**key** a string expression defining the letters required for partial string match. Upper case letters define the minimum required spelling. Key is used to validate the argument string.

**if_true** a string expression whose value the function takes if the test is successful.

**if_false** an optional string expression whose value the function takes if the test fails. If this argument is omitted, the function takes on the value of a null string.

**Examples:**

Ask the user for a Yes/No response and compare the reply with the key "Yes". Require the user to type at least "Y":

```
eFT> ask -prompt "Yes/No? " reply
eFT: Yes/No? y
eFT> text {cmp(reply,"Yes","YES","NO")}
eFT: YES
```

Ask the user for a Yes/No response and compare the reply with the key "YES". Force the user to type the entire word "YES":

```
eFT> ask -prompt "YES/No? " reply
eFT: YES/No? yes
eFT> text {cmp(reply,"YES","YES,"NO")}
eFT: YES
```

In order to not compare to the key, do the same commands, but reply to the prompt with text that doesn't meet the minimum spelling requirements:

```
eFT> ask -prompt "YES/No? " reply
eFT: YES/No? y
eFT> text {cmp(reply,"YES","YES","NO")}
eFT: NO
```

## DATE Function

The DATE function returns the system date of the local host.

**Format:**

```
date([number])
```

Where:

**number** this is a number that specifies the format of the date. For the DATE function:

**0** = WWW MMM DD, YYYY

**1** = YYMMDD

where **W** = Weekday; **M** = Month; **D** = Day; **Y** = Year. Specifying a value other than 0 or 1 returns a null value. The default is 0.

**Examples:**

Display the system date:

```
yellowstone> text Today is {date()}
14:32:39 MVS:Today is Thu Jun  6, 2019
yellowstone>
```

Display the system date in YearMonthDay format:

```
yellowstone> text Today is {date(1)}
14:33:52 MVS:Today is 190606
yellowstone>
```

## DEC and INC Functions

**DEC** subtract one from a numeric string expression and return the result.
**INC** adds one to a numeric string expression and return the result.

**Format:**

```
dec(number)
inc(number)
```

Where:

**number** the number to be operated on.

**Examples:**

Display the results of incrementing "5":

```
eFT> text Increment 5 = {inc(5)}
eFT: Increment 5 = 6
```

Assume the variable "CNT" exists and has a value of "12". The value of "CNT" minus one can be displayed as follows:

```
eFT> text New cnt = {dec(cnt)}
eFT: New cnt = 11
```

This procedure did not change the actual value of "CNT"; it only displayed the decremented value. To decrement the value of the variable "CNT", use the **SET VARIABLE** command as shown:

```
eFT> text cnt = {cnt}
eFT: cnt = 12
eFT> set variable cnt {dec(cnt)}
eFT> text cnt = {cnt}
eFT: cnt = 11
```

## DFN and NDF Functions

The **DFN** function tests if a variable is defined. The **NDF** function tests if a variable is not defined.

**Format:**

```
dfn(variable , if_true [ , if_false ])
ndf(variable , if_true [ , if_false ])
```

Where:

| | |
|---|---|
| **variable** | a string variable that is to be tested. A string is undefined if it has no value (i.e., null string). |
| **if_true** | a string expression whose value the function takes if the test is successful. |
| **if_false** | an optional string expression whose value the function takes if the test fails. If this argument is omitted, the function takes on the value of a null string. |

**Examples:**

To find out if a variable is defined, such as the variable "COUNT":

```
eFT> text {dfn(count,"YES","NO")}
```

If "COUNT" was defined, the command would result in the following display:

```
eFT: YES
```

Set the input prompt to be the remote host variable if it is defined; otherwise use the string literal "NTXeFT":

```
eFT> set input prompt {} {dfn(host:remote,host:remote,"NTXeFT")}>
```

If the remote host name is "BLUESKY", the prompt would be:

```
BLUESKY>
```

**NDF** can also be used to set the input prompt to be the remote host variable or to the string literal "NTXeFT":

```
eFT> set input prompt {} {ndf(host:remote,"NTXeFT",host:remote)}>
eFT>
```

If the remote host name is "BLUESKY", the prompt would be:

```
BLUESKY>
```

Define an alias "PRINT" to output the first parameter passed to it, or to output "no parameter" if no parameter is passed.

```
eFT> set alias print {} {ndf(1,"text no parameter","text {1}")}
```

Execute the alias with no parameters:

```
eFT> print
eFT: no parameter
```

Now execute the alias with the parameters "this text":

```
eFT> print this text:
eFT: this
```

## ENCrypt Function

The **ENCrypt** function can be used to encrypt user passwords which will be used by NetEx/eFT to establish connections to remote hosts. All other data exchanges will be in clear text format. Use a secure connection to encrypt all interactions between the host and the remote.

Sites often store remote system access information in files to be included on the **CONNECT** command during NetEx/eFT execution. To prevent from storing cleartext passwords in files, you can encrypt a password and store the encrypted form of the password in the file. The encrypted form of the password is useless to an individual for gaining access to a remote system outside of NetEx/eFT. Within NetEx/eFT, the encrypted form of the password is valid only when decrypted internally by NetEx/eFT using the local host username as a secondary encryption key.

**Format:**

```
ENCrypt("password" [,"username"])
```

Where:

**password** (STRING) specifies the remote system password you want to encrypt. The encrypted form of this password is returned by **ENCRYPT** and should be stored in a file for later use with the **CONNECT** command. You may leave this parameter off the command line and be prompted for it.

**username** (STRING) specifies the local host username which represents the username associated with the local NetEx/eFT process that will issue the **CONNECT** command with the encrypted password. This local host username is used as a secondary encryption key for the specified password. When NetEx/eFT is later run on the local system, either interactively or in batch, it queries the operating system for the username of the process. NetEx/eFT then uses this username as one of its keys in decrypting the password. The value for "username" above must be entered in uppercase in order to match the username value later returned by MVS/zOS/Tandem and VMS.

**Examples:**

Assume a password of "special" is to be used for the remote user and the local user that will be issuing the connect is "TEST3". Display the value of the encrypted password:

```
eFT> text {enc("special","TEST3")}
eFT: *2952a3d6a882e2311
```

Assume a password of "special" is to be used for the remote user and that NO local user will be specified. Display the value of the encrypted password:

```
eFT> text {enc("special")}
eFT: *19401e92a796d093a
```

## ENV Function

The **ENV** function returns the value of the local host environment variable if the local host supports such variables. If local host environment variables are not supported or if the variable is not defined, a null string is returned. In IBM z/OS NetEx/eFT, the value of an environment variable is obtained from the TSO/E CLIST variable pool.

**Format:**

```
env(variable)
```

Where:

**variable** a local host environment variable. This variable may be upper-/lower-case sensitive. The definition of such a variable depends on the local host.

**Examples:**

Assume a variable "bite" is defined to be "apple" in the local host's environment. Display the value of the host environment variable with **ENV**:

```
eFT> text {env("bite")}
eFT: apple
```

To return the host environment variable "SYSUID", the TSO/E user ID:

```
eFT> text {env("sysuid")}
eFT: GUEST1
```

## EQS and NES Functions

The **EQS** function tests if the first string is equal to the second. The **NES** function tests if the first string is not equal to the second. These tests are case sensitive, so "Ab" is not equal to "AB".

**Format:**

```
eqs(string1, string2, if_true [ , if_false])
nes(string1, string2, if_true [ , if_false])
```

Where:

| | |
|---|---|
| **string1, string2** | these can be variable or literal string expressions. |
| **if_true** | a string expression whose value the function takes if the test is successful. |
| **if_false** | an optional string expression whose value the function takes if the test fails. If this argument is omitted, the function takes on the value of a null string. |

**Examples:**

Assume when connecting to most systems with the username "GUEST", a password is usually not required. Set the username variable "usr" to the text "person". Then, if the username is NOT "GUEST", ask for a password:

```
eFT> set var usr person
eFT> {nes(upper(usr),"GUEST","ask -prompt ""Password? "" pass")}
Password? _______
```

Set the username variable "usr" to the text "guest". Then, if the Username is NOT "GUEST", ask for a Password:

```
eFT> set var usr guest
eFT> nes(upper(usr), "GUEST", "ask -prompt " "Password? " "pass")}
eFT>
```

In most cases, when using the username "GUEST" to make a connection, if a password is required (and known) it can be automatically set to the correct input. To try this, set the username variable "usr" to the text "guest". Then, if the username is "GUEST", set the password to "NETEX":

```
eFT> set var usr guest
eFT> {eqs(lower(usr), "guest", "set var pass netex")}
eFT>
```

## EXT Function

The EXT function extracts and returns a bounded sequence of characters from a string.

**Format:**

```
ext(string, number1, number2)
```

Where:

| | |
|---|---|
| **string** | a string expression in the form of a variable, literal, or function. |
| **number1, number2** | the lower and upper boundary limits for the characters to be extracted from **string**. Parameter values less than or equal to zero are interpreted relative to the end of the string. |

**Example:**

Display the sequence "CDE" from the string "ABCDEF":

```
eFT> text {ext("ABCDEF",3,5)}
eFT: CDE
```

The same sequence ("CDE") may be displayed by using parameter values relative to the end of the string, as shown:

```
eFT> text {ext("ABCDEF",-3,-1)}
eFT: CDE
```

Display only the hours and minutes of the system time:

```
eFT> text Time: {ext(time(),1,5)}
eFT: Time: 14:27
```

Define the input prompt to display "NTXeFT" and the version number of NetEx/eFT extracted from "VERSION:LOCAL":

```
eFT> set input prompt {} NTXeFT {ext(version:local,1,3)}>
NTXeFT 5.5>
```

## INDEX Function

The INDEX function returns the position of the second string specified within the first string. The function returns zero if the second string is not found.

**Format:**

```
index(string1, string2)
```

Where:

**string1, string2** string expressions in the form of a variable, literal, or function.

**Examples:**

Display the position of the sequence "CDE" within "ABCDEF":

```
eFT> text {index("ABCDEF", "CDE")}
eFT: 3
```

Find the position of the month "March" in a string containing a list of the months:

```
eFT> text {Index("JanFebMarAprMayJunJlyAugSepOctNovDec", "Mar")}
eFT: 7
```

The following is an example of an **index()** search that failed to find the second string within the first:

```
eFT> text {index("abcdef","cat")}
eFT: 0
```

## LEN Function

The **LEN** function returns the count of characters that make up the specified string.

**Format:**

```
len(string)
```

Where:

**string** a string expression in the form of a variable, literal, or function.

**Examples:**

Display the number of characters in "ABCDE":

```
eFT> text {len("ABCDE")}
eFT: 5
```

Display the length of the results of the **DATE()** string function:

```
eFT> text Length = {len(date())}
eFT: Length = 16
```

Display only the year portion of the system date by subtracting three from the length of the date and then extracting the last four characters:

```
yellowstone> text Year: {ext(date(),sub(len(date()),3),len(date()))}
14:35:28 MVS:Year: 2019
yellowstone>
```

A simpler form of the example above is:

```
yellowstone> text Year: {ext(date(),-3,0)}
14:36:20 MVS:Year: 2019
yellowstone>
```

## Logical Operations

The following is a list of the operators for numerical equivalence tests:

**EQ** tests if the first number is equal to the second number.
**NE** tests if the first number is not equal to the second number.
**LT** test if the first number specified is less than the second.
**GT** test if the first number specified is greater than the second.
**LE** test if the first number specified is less than or equal to the second.
**GE** test if the first number specified is greater than or equal to the second.

Format:

```
eq(number1, number2, if_true [, if_false ])
ne(number1, number2, if_true [, if_false ])
lt(number1, number2, if_true [, if_false ])
gt(number1, number2, if_true [, if_false ])
le(number1, number2, if_true [, if_false ])
ge(number1, number2, if_true [, if_false ])
```

Where:

| | |
|---|---|
| **number1, number2** | numbers to be compared. |
| **if_true** | a string expression whose value the function takes if the test is successful. |
| **if_false** | an optional string expression whose value the function takes if the test fails. If this argument is omitted, the function takes on the value of a null string. |

**Examples:**

Ask the user to enter a number between 1 and 10:

```
eFT> ask -prompt "Enter a # (1-10): " num1
eFT: Enter a # (1-10): 10
```

Check if the number is less than or equal to 10:

```
eFT> text {le(num1,10, "Good", "Bad")}
eFT: Good
```

Check the number for proper entry as defined in the first example, if it is between 1 and 10:

```
eFT> text {ge(num1,1,le(num1,10, "Good", "Bad"), "Bad")}
eFT: Good
```

## LOWER and UPPER Functions

The **UPPER** function returns the upper-case equivalent of a string expression, all characters except lower-case characters are left untouched.

The **LOWER** functions return the lower-case equivalent of a string expression, all characters except upper-case characters are left untouched.

**Format:**

```
upper(string)
lower(string)
```

Where:

**string** a string expression in the form of a variable, literal, or function.

**Examples:**

Display the lower-case equivalent of "ABCdef":

```
eFT> text {lower("ABCdef")}
eFT: abcdef
```

Display the upper case equivalent of "ABCdef":

```
eFT> text {upper("ABCdef")}
eFT: ABCDEF
```

Display the system date in lower case:

```
eFT> text Today is: {lower(date())}
eFT: Today is: thu jun  6, 2019
```

To display the Weekday in upper case:

```
eFT> text The Day is: {ext(upper(date()),1,3)}
eFT: The Day is: THU
```

To set a predefined NetEx/eFT variable "password" to its lower-case equivalent:

```
eFT> set var password {lower(password)}
eFT>
```

## MSG Function

The **MSG** function returns the information of the last message. **MSG** is most often used to tailor the NetEx/eFT output message format.

**Format:**

```
msg(component [, facility])
```

Where:

**component** the code for the type of message data to return. Valid message components are: **Text**, **Facility**, **Code**, **Severity**, **Retry**, or **Purge**.

- **Text** requests the text from the message.
- **Facility** requests the source of the message: NETEX, UA, UAxxx (where xxx represents the host product code), SI, SIxxx, MUX, MUXxxx, or the operating system mnemonic for the host generating the error.
- **Code** requests the message number from the facility.
- **Severity** requests the severity of the message: E, W, or I for Error, Warning, and Informational respectively.
- **Retry** requests whether the network error can be retried (not fatal). Results are either Y, for can-be-retried, or N, for cannot-be-retried.
- **Purge** will purge the message stack. One may find it useful to ensure that the next attempt to read a message resulted in a message from the last command, in this case the user will need to Purge the message stack.

**facility** to get the message from a certain facility: NETEX, UA, UAxxx (where xxx represents the host product code), SIxxx, MUXxxx, or the operating system mnemonic for the host generating the error. The default is UA.

**Examples:**

Display the facility of the last error message:

```
eFT> text The Last Error Message came from: {msg("f")}
eFT: The Last Error Message came from: UA
```

Display the text of the last error message from NetEx/eFT:

```
eFT> text The Last Error Message was: {msg("t")}
eFT: The Last Error Message was: Invalid command 'oops'
```

Display the code of the last error message from NetEx/eFT:

```
eFT> text The Last Error Code was: {msg("c")}
eFT: The Last Error Code was: 4708
```

Set the **OUTPUT FORMAT** qualifier to display only the message text when an error occurs:

```
eFT> set output format {} {msg("text")}
```

## PARAMS Function

The **PARAMS** function substitutes the positional parameters specified. It is important to note that quoted parameters remain quoted and are considered one whole parameter - regardless of imbedded spaces.

**Format:**

```
params(number1, number2 [, char])
```

Where:

| | |
|---|---|
| **number1, number2** | positional parameters number1 through number2 for substitution. Use number2 = "0" to mean "the rest". |
| **char** | is the optional parameter separator to use. The default is a space. |

**Examples:**

Define an alias "PRINT" to display the parameters passed to it:

```
eFT> set alias print {} text {params(2,3)}
```

Execute the alias with no parameters:

```
eFT> print
eFT:
```

Now execute the alias with the parameters: this text string

```
eFT> print this text string
eFT: text string
```

Execute the alias with the parameters: this "is a" test:

```
eFT> print this "is a" test
eFT: "is a" test
```

## SLEEP Function

The SLEEP function causes NetEx/eFT to pause or "sleep" for a specified number of seconds. This process is not interruptible. This function results in a null string.

**Format:**

```
sleep(number)
```

Where:

**number** number of seconds to pause/sleep.

**Examples:**

To pause/sleep a NetEx/eFT session for ten seconds:

```
eFT> {sleep(10)}
eFT>
```

If an error occurs during a connect command, the alias **RECONnect** will go to sleep for thirty seconds and then attempt to connect again:

```
eFT> set alias RECONnect {} set input continue on !
More>> start: !
More>> con {params(1,0)} !
More>> eqs(status(),"S","exit")} !
More>> {sleep(30)} !
More>> goto start
```

## STATUS Function

The **STATUS** function returns the single status character of the previous command: S = Success, E = Error. Successful execution of the following commands leaves the previous command status intact: **CONTINUE**, **EXIT**, **GOTO**, **QUIT**, and **TEXT**. This gives the user the ability to position within a script (for example, to an error processing section) without clearing the status from previous failures. If any of these commands fail, an error status is set. A status return specified by **EXIT** or **QUIT** (for example, **EXIT ERROR**) will override previous command status.

The status returned from a **LOCAL** or **REMOTE** command is the result of the NetEx/eFT **LOCAL** or **REMOTE** command and not the completion code set by the resulting host command. To check the status of the host command, use the variables {**status:local}** or **{status:remote}**. If **LOCAL** or **REMOTE** command execution is not supported on the requested host, the **status()** function will be set to 'E'.

**Format:**

```
status()
```

**Examples:**

To display the output of the status function:

```
eFT> text {status()}
eFT: S
```

To display the status of the last command:

```
eFT> text Last Command {eqs(status(),"S","Succeeded","Failed")}
eFT: Last Command Succeeded
```

Define the input prompt to signal a message when an error has occurred:

```
eFT> set input prompt {} {nes(status(),"S","Error ")}
eFT>
```

With the above definition, issue an invalid command and then a valid command to test the new prompt:

```
eFT> oops
eFT: Invalid command 'oops' (UA-4708).
Error
eFT> text Correct the Prompt
eFT: Correct the Prompt
eFT>
```

## TIME Function

The TIME function returns the system time of the local host.

**Format:**

```
time([num])
```

Where:

**num** this is a number that specifies the format of the time. For the **TIME** function:

**0** = HH:MM:SS

**1** = HHMMSS

Where H = Hours; M = Minutes; S = Seconds. Specifying a value other than 0 or 1 returns a null string. Default is 0.

**Examples:**

Output the time without the colon separator:

```
eFT> text The Time is: {time(1)}
eFT: The Time is: 142448
```

Output the time:

```
eFT> text The Time is: {time()}
eFT: The Time is: 14:24:48
```

Redefine the input prompt to prompt with the system time:

```
eFT> set input prompt {} {time()}>
14:25:04>
```

The empty braces ({}) in the above example are needed to disable string substitution until each prompt is displayed. The new system time is then evaluated each time the prompt is displayed.

## Disabling String Substitution

When NetEx/eFT sees "{string}" on a command line, it immediately tries to perform string substitution on the string. To disable string substitution, place an empty "{}" on the command line prior to the string substitution syntax. The typical place to do this is during an alias definition. For example:

```
eFT> set alias put send {} {sourcefile} {destinationfile}
```

Upon seeing the empty "{}" before the string substitution syntax "{sourcefile}" and "{destinationfile}", NetEx/eFT knows to not substitute the values of "sourcefile" and "destinationfile" at this time. The resulting definition for alias **PUT** is:

```
eFT> show alias put
eFT: PUT ............... send {sourcefile} {destinationfile}
```

If the empty "{}" had not been included during the definition above, NetEx/eFT would have replaced "{sourcefile}" and "{destinationfile}" by their current values at the time the alias was defined. If they were undefined, they would have been replaced by the null string. The goal of an alias is usually to replace the value of the variable at the time the alias is run, not when it is defined.

The empty "{}" is used as a toggle to turn string substitution on and off. In the following example, the first occurrence of "{}" turns off string substitution, which results in "{sourcefile}" not being replaced by its value. The second occurrence of "{}" turns string substitution back on. This results in "{destinationfile}" being replaced by its current value (assume it is "dest.new"):

```
eFT> set alias put send {} {sourcefile} {} {destinationfile}
```

The resulting alias definition looks like:

```
eFT> show alias put
eFT: PUT ............... send {sourcefile} dest.new
```

In this case, the alias **PUT** becomes a send command where the destination file is always "dest.new". Since "{}" is used as a toggle, it should only appear once within an alias definition (including multi-command aliases) when string substitution is to be ignored for all variables declared.

## Nested String Substitution

The string substitution syntax also allows for nested substitution. Nested substitution provides for embedding string substitution syntax within string substitution. Figure 2 is a representation of nested substitution.

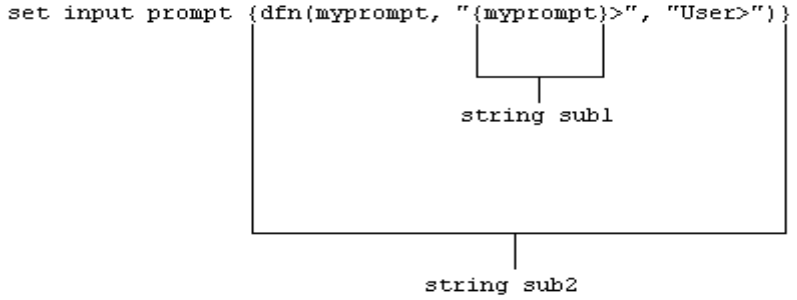



**Figure 2. Nested String Substitution**

The example above sets the NetEx/eFT input prompt to either the value of variable "myprompt" (if it is defined), or else to the string "eFT>". Since there exists nested string substitution, NetEx/eFT first processes the

innermost one (labeled “string sub1” above) to evaluate the variable “myprompt”. The double quotes outside of “{myprompt}” turns the resulting value into a string literal that is then used as the second argument to the **dfn()** function. Once this is done, NetEx/eFT processes the outermost string substitution syntax (labeled “string sub2” above).

All string substitution processing is performed by NetEx/eFT from the inside out. This is important to keep in mind when creating such things as custom prompts or scripts. Since the inside string substitution syntax is processed first, it is treated as a separate entity in and of itself. This is significant because it affects the use of double quotes for string literals. Normally double quotes must be escaped when they are used within another set of double quotes for NetEx/eFT to take them literally. However, if the outer set of double quotes is not part of the immediate string substitution syntax containing the inner set of double quotes, then the inner set of double quotes should not be escaped. The following example illustrates this point.

```
eFT> ask -prompt "{upper("enter your name")}" name
```

Since the outer set of double quotes (“{...}”) is outside of the string substitution syntax, the inner set should not be escaped. This is because the string substitution syntax is processed first, resulting in the non-quoted string “ENTER YOUR NAME”. The outer set of quotes then is applied to that string and the remainder of the command is processed.

# Developing NetEx/eFT Scripts Using Input Files and Aliases

NetEx/eFT was designed to be very easy to use for all types of users. The commands are simple, and the syntax is straightforward. However, it is often the case that a site wants to customize the NetEx/eFT interface to be even more simple or familiar for its users. This customizing is generally done by more sophisticated NetEx/eFT users then handed back to the general user base. This section addresses the areas important to developing NetEx/eFT scripts.

A script can be in the form of an input file or an alias; NetEx/eFT treats them the same internally. The difference is in the way they are defined. Input files are created using a standard text editor. Aliases are created using the **SET ALIAS** command and require special command line syntax. The examples that are given in this section, although they apply to all types of scripts, are generalized to emphasize the topic of discussion and do not include the special syntax required for creating aliases. The reader should be familiar with NetEx/eFT command line processing, most notably, the sections on “Special Characters” on page 63 and “NetEx/eFT String Substitution” on page 64.

## NetEx/eFT Input Files

NetEx/eFT input files or input scripts give users the ability to write powerful program-like procedures that can be run on several different host types, without regard to host-specific command language differences. A NetEx/eFT input file or input script is a file that contains a list of NetEx/eFT commands. Input scripts are created using any standard text editor.

Assume for the following examples, that there exists an input script called “HOSTDIR”, shown below, which connects to a predefined host named “BETA”, issues a **REMOTE DIRECTORY** command, and then disconnects.

```
* Input script HOSTDIR - Give directory listing of
*                        remote host named BETA.
*
connect beta default test
remote dir
disconnect
```

There are three ways to make use of an input script:

1. With the **INPUT** command. The **INPUT** command tells NetEx/eFT to read and execute the NetEx/eFT commands in the input file given. For example:

   ```
   eFT> input hostdir
   ```

   This command tells NetEx/eFT to look for an input script with the file specification "HOSTDIR" and then read it line by line, executing commands along the way.

2. Using the **INPUT** qualifier **SEARCH** - when NetEx/eFT reads a command from the command line, it first looks for an alias by that name and translates it if found. If the command is not an alias, NetEx/eFT determines if it is a NetEx/eFT command. If not, it looks at the **INPUT** qualifier **SEARCH**. If that qualifier is defined, NetEx/eFT uses the **SEARCH** path to find an input file by the name of the command it read. If an input file is found, NetEx/eFT reads and executes it. If no input file is found, NetEx/eFT issues an "Invalid command" error. Therefore, the second way to use the "HOSTDIR" input script is to first define the **INPUT SEARCH** qualifier (see the "INPUT" Command in "Command Descriptions" on page 117), and then type "HOSTDIR" on the command line:

   ```
   eFT> hostdir
   ```

   Typing "HOSTDIR" here gives the appearance that "HOSTDIR" is a NetEx/eFT command. The **INPUT SEARCH** qualifier can also contain **SEARCH** keywords (SITE) and (USER). Refer to "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106 for more information

3. On the NetEx/eFT command line - The third option is to specify the input file on the NetEx/eFT command line when NetEx/eFT is invoked. This option is most often used when running NetEx/eFT within a batch job. It simply tells NetEx/eFT to read and execute all of the commands within the input script and then exit NetEx/eFT. For more information, see "Running an IBM z/OS NetEx/eFT Client in Batch" on page 110 and "Invoking NetEx/eFT on IBM z/OS" on page 9.

## Echoing Input Scripts to the Terminal

In order to have NetEx/eFT display the **INPUT** commands as they are executed, the user must turn on input echo with the **SET INPUT** command:

```
* Echo all commands as they execute
*
set input echo on
```

This command can be issued interactively, prior to the **INPUT** command, or given as the first command in the input file. **INPUT ECHO** will echo each command as it appears prior to string substitution. To display each input command after string substitution, turn on the **INPUT VERIFY** qualifier as:

```
* Echo all commands after string substitution
*
set input verify on
```

Refer to the INPUT command in "Command Descriptions" for more details on these qualifiers. The section "Debugging a NetEx/eFT Alias or Input Script" on page 103 also addresses the **VERIFY** qualifier.

## Displaying Output and Accepting Input within a Script

The **TEXT** command is used to display output within a NetEx/eFT session. For example, the commands in a script to display a welcome message to a user could be:

```
text  ********************************
```

```
text       Welcome to EFT!
text  ********************************
```

A **TEXT** string can also include NetEx/eFT string substitution syntax "{...}". Therefore, string literals, string variables, or string functions can be substituted within a **TEXT** command line to provide additional information. For example, to enhance the example above, the string function **DATE()** could be used, along with qualifier values **VERSION** and **DIRECTORY** from the **LOCAL** environment:

```
text  ************************************************
text     Welcome to EFT Version {version:local}.
text
text                    {date()}.
text
text     Your current directory is: {directory:local}.
text  ************************************************
```

The **ASK** command can be used to make scripts more friendly for novice users. For example, a "TRANSFER" script could be defined that would prompt a user for worthwhile information then execute the commands on the user's behalf:

```
* This is a sample TRANSFER script
*
ask -prompt "Transfer being made to what host? " host
ask -prompt "User ID on host {host}? " uid
ask -secure -prompt "Password for user {uid}? "  pw
ask -prompt "File to be transferred? " file
connect -quiet {host} {uid} {pw}
send -quiet {file}
text **********************
text File '{file}' has been transferred to {host}.
text **********************
disconnect -quiet
```

To execute the script (if appropriate setup was done by the site administrator), the user would type only "TRANSFER":

```
eFT> transfer
Transfer being made to what host? BLUESKY
User ID on host BLUESKY? guest
Password for user guest? _____
File to be transferred? tmpfile
**********************
File 'tmpfile' has been transferred to BLUESKY.
**********************
eFT>
```

## Passing Parameters to a Script

Parameters may be passed to a NetEx/eFT script. These parameters are referred to as positional parameters because they are identified by their position on the command line. For example, the following **INPUT** command passes the input file "SETUP" three positional parameters, the first positional parameter is "HOSTA", the second is "SMITH", and the third is "JOHN":

```
eFT> input setup hosta smith john
```

An input file can be passed several positional parameters. Each parameter is identified in the input file by its position number in braces ("{1}", "{2}", "{3}", etc.). In the example above, "HOSTA" is represented by "{1}" in the input script "SETUP", "SMITH" is "{2}", and "JOHN" is "{3}". In general, the mapping is:

```
{0}  ...  positional parameter zero - the entire parameter string
{1}  ...  positional parameter one
{2}  ...  positional parameter two
 .
 .
 .
{n}  ...  positional parameter n
```

Positional parameters are used with an input script using the syntax described above. For example, the file "SETUP" may contain the three lines:

```
text {3} {2} is attempting to connect to host {1}
connect {1} {2} {3}
text hello {3} {2}.  How are you?
```

When the **INPUT** command is issued, NetEx/eFT performs string substitution. After string substitution, the command lines appear as:

```
eFT> input setup hosta smith john
text john smith is attempting to connect to host hosta
connect hosta smith john
text hello john smith.  How are you?
```

To pass multiple words or strings with embedded blanks as a single parameter, enclose them in double quotes. Refer again to the example above. If the third parameter was "JOHN HENRY" instead of just "JOHN", the string "JOHN HENRY" would have to be enclosed in double quotes:

```
eFT> input setup hosta smith "john henry"
```

Now, parameter 1 is "HOSTA", parameter 2 is "SMITH", and parameter 3 is "JOHN HENRY".

Up to this point positional parameter passing and input prompting has been discussed. The next section combines these features by making use of string functions.

## Using String Functions within a NetEx/eFT Script

In order to gain even more flexibility, a script can be designed to use positional parameters if they are passed, and to prompt for input in the absence of positional parameters. This added feature of scripting makes use of NetEx/eFT string functions. (See the section entitled "String Functions" on page 68 for a more detailed discussion.)

In the sample below, the string function **ndf()**, (if Not Defined), is used. **ndf()** tests its first argument (variable 1, 2, 3, or 4) to see if it's defined. If it's not, **ndf()** executes its second argument (**ASK -prompt** ...). If argument 1 is defined, **ndf()** executes argument three (optionally left out). The new definition of the "TRANSFER" script is:

```
* This is a sample TRANSFER script that uses String Functions
*
{ndf(1, "ask -prompt ""Transfer being made to what host? "" 1")}
{ndf(2, "ask -prompt ""User ID on host {1}? "" 2")}
{ndf(3, "ask -secure -prompt ""Password for user {2}? ""  3")}
{ndf(4, "ask -prompt ""File to be transferred? "" 4")}
connect -quiet {1} {2} {3}
send -quiet {4}
text **********************
text File '{4}' has been transferred to {1}.
text **********************
disconnect -quiet
```

The logic of the script when invoked is:

- If no parameters are on the "TRANSFER" command line, prompt the user for all four pieces of information and read them into variables 1, 2, 3, and 4.
- If one parameter is passed, use it as variable 1 ("Host"), and prompt for the other three.
- If a second parameter is passed on the TRANSFER command line, use it as variable 2 ("User ID"), and prompt for the third and fourth variables.
- If three parameters are passed on the command line, read them into variables 1, 2, and 3 ("Password") respectively and prompt only for the fourth variable.
- If four parameters are passed on the command line, read them into variables 1, 2, 3 and 4 ("file") respectively, and do not prompt at all.
- Finally, execute the remaining commands, substituting the variables or parameters as needed.

The following is an example execution of the "TRANSFER" script with two parameters passed to it:

```
eFT> transfer BLUESKY guest
Password for user guest? _____
File to be transferred? tmpfile
**********************
File 'tmpfile' has been transferred to BLUESKY.
**********************
eFT>
```

Since two parameters were passed, the script prompted only for the last two. Scripts like this one are especially useful when there is a need to run both interactively and in batch. Batch jobs require all the parameters to be defined since they cannot prompt for user input.

## Using NetEx/eFT Labels and GOTOs

To make NetEx/eFT scripts even more powerful, users can merge string functions with **GOTO** processing. The **GOTO** command instructs script processing to continue at the specified label, either backwards or forwards. The format is:

| Command | Parameters |
|---|---|
| GOTO | Label |

Where:

**label** is an alphanumeric string from one to twenty characters (including underscores) in length.

All labels are case sensitive and must appear somewhere within the current input level. That is, if the **GOTO** appears within an input script, the matching label must also be in that input script. If the **GOTO** appears in an interactive input level, the matching label must also be found within that interactive level. The format of a command line that contains a label is:

```
label: [command]
```

The colon immediately following the label is required. The label can appear on a line by itself, or it may be followed by a valid NetEx/eFT command or alias. The following is an example of a simple loop alias:

```
*  Sample GOTO/Label script.  Send 5 files
*  having the names FILE1 thru FILE5.
*
```

```
set variable count 1
LOOP:
send FILE{count}
set variable count {inc(count)}
{le(count, 5, "goto LOOP")}
*
text All files sent.
```

The variable count is first initialized to 1. The next line simply declares a label called "LOOP". The **SEND** command is then issued for a file named "FILE*x*" where *x* is the current value of count. Following that, the variable count is reset to its value plus one (incremented by one). Finally, a check is made on the value of count. If it is less than or equal to 5, the **GOTO LOOP** command is substituted as the next NetEx/eFT command and processing branches up to label **LOOP**. If count exceeds 5, processing falls through to the next command outside of the loop.

It is important to remember that NetEx/eFT treats labels as case sensitive. Therefore, one must make sure that a label specified on a **GOTO** command matches the actual label's case exactly. Duplicate labels (labels that have the same name with identical case) are considered an error.

Whenever a **GOTO** or a label is encountered during command line processing, all future commands get stored internally within NetEx/eFT. The number of commands that can be stored is limited by the amount of available memory allocated for the process which varies from machine to machine. The best practice is to avoid letting scripts that contain **GOTO**s become too large.

Since scripting is really an interpretive command language, NetEx/eFT must parse each command as it executes it. Therefore, an error within a script will not be caught until the script is run and the erroneous condition is encountered. A missing label, for example, will result in the entire script being read before an error message is given.

**GOTO** and labels may also appear at the interactive session level. Refer to the **GOTO** command in "Command Descriptions" on page 117 of this manual for further information.

## Using the ON (ERROR/INTERRUPT) Command

A useful command for building more robust scripts is the ON command. ON has the format:

| Command | Parameter |
|---|---|
| ON | exception [action] |

Where:

**exception** is any one of the following:

| | |
|---|---|
| **ERRor** | on NetEx/eFT error perform action |
| **INTerrupt** | on keyboard interrupt perform action |
| **LOCal_error** | on LOCAL command error perform action |
| **REMote_error** | on REMOTE command error perform action |

**action** is any valid NetEx/eFT command or alias, the most likely of which being:

| | |
|---|---|
| **CONTinue** | ignore the exception |
| **EXIT** | exit the current NetEx/eFT session |
| **GOTO** | GOTO a specified label |
| **INPUT** | input a specified NetEx/eFT script |
| **TEXT** | display a message |

**<none>** turn off the specified exception

If action is more than one command, the results are unpredictable.

The **ON** command allows a user to catch any one of the exceptions listed above and perform a predefined action. It is most useful within NetEx/eFT scripts for tailoring exception handling. **ON** commands generally appear at the beginning of a NetEx/eFT script since they set up actions to be taken on future processing within that script. For example, the following script catches any keyboard interrupt (initiated by the user), and automatically causes the session to terminate:

```
*   Cause keyboard interrupt to exit session
*
on interrupt exit
connect hosta guest netex
send src_file
disconnect
```

The **ON INTERRUPT** exception as shown above, establishes an alternative action to be taken in the case of a user generated keyboard interrupt. By default, without an **ON INTERRUPT** specified, NetEx/eFT terminates all input levels (if within nested input scripts) and returns to the interactive level. If the **INPUT CONTINUE** qualifier was set, NetEx/eFT terminates only the current input level and continues processing in the next level up.

The **ON ERROR** exception establishes an alternative action to be taken when a NetEx/eFT error occurs. Without an **ON ERROR** specified, NetEx/eFT terminates all input levels (for nested input scripts) and begins processing at the interactive level. If the **INPUT CONTINUE** qualifier was set, NetEx/eFT displays the error but continues processing the next command. The **ON ERROR** command allows for more flexibility on an error condition.

The **ON LOCAL_ERROR** and **ON REMOTE_ERROR** exceptions give the user the ability to take special action when a **LOCAL** or **REMOTE** host command fails. On hosts that support it, a special **LOCAL** or **REMOTE STATUS** qualifier will be set reflecting that host's error code for the error condition. Normally, without **ON LOCAL_ERROR** or **ON REMOTE_ERROR** specified, NetEx/eFT just ignores host errors and continues processing the next command.

For further details about the **ON** command, see "Command Descriptions" on page 117.

## Checking Command Status

The **status()** function allows the user to write scripts that check the status of the previous command and take action accordingly. A simple example appears below:

```
*
*     Print status of a single file transfer.
*
set input continue on
send -quiet src_file
text Transfer {eqs(status(), "S", "Successful.", "Failed.")}
```

The example above demonstrates how **status()** can be used to print the results of the previous command (in this case **SEND**). **status()** returns either an "S" for Success, or an "E" for Error. Since it is a string function, the result is a string. **status()** can be combined with any other string function or NetEx/eFT command to enhance script processing. Above, it is used with the **eqs()** function to check the results which are then printed by the **TEXT** command.

Refer to the section on string functions for more details and examples of the **status()** function.

## Creating NetEx/eFT Aliases

Aliasing is simply another form of NetEx/eFT scripting. Although the previous sections discussed scripting in the context of **INPUT** files, everything described applies to NetEx/eFT aliases. Aliases can be thought of as NetEx/eFT input files. In fact, a multi-command alias (discussed later) is treated in the same way. The difference between a multi-command alias and an input script is the way in which they are defined. As mentioned earlier, input scripts are defined using a standard text editor. Aliases are defined using the **SET ALIAS** command.

The alias capabilities of NetEx/eFT provide a means of creating a custom command set that can be used by all users or a group of users. An alias is simply a new name for a NetEx/eFT command or set of commands. The following example shows how to create a simple alias (or new command) called "FETCH" that is equivalent to the **RECEIVE** command:

```
eFT> set alias fetch receive
```

To display the definition of the new alias, use the **SHOW ALIAS** command:

```
eFT> show alias fetch
eFT: FETCH ............. receive
```

The alias name "FETCH" appears on the left and its translation or definition appears on the right. Now to transfer a file from the remote host to the local host, either the **RECEIVE** command or the "FETCH" alias can be used. All qualifiers and parameters for the **RECEIVE** command are also valid for "FETCH" since NetEx/eFT just maps "FETCH" to **RECEIVE**. The **RECEIVE** command is executed, but as far as the user is concerned, the command executing is "FETCH". Therefore, the following are equivalent:

```
eFT> fetch -quiet -mode character sourcefile
```

and,

```
eFT> receive -quiet -mode character sourcefile
```

Aliases can be more complex than a single mapping of "FETCH" to **RECEIVE**. For example, the "FETCH" alias can be defined to include parameters or qualifiers as part of its definition. Below is an example of the "FETCH" alias that includes some **RECEIVE** qualifiers in its definition:

```
eFT> set alias fetch receive -quiet -mode character
```

To display the new definition alias "FETCH", the **SHOW ALIAS** command is again used:

```
eFT> show alias fetch
eFT: FETCH ............. receive -quiet -mode character
```

Now every time "FETCH" is invoked, the **QUIET** qualifier is turned on and the **MODE** is set to **CHARACTER**. Therefore, the following are equivalent:

```
eFT> fetch sourcefile
```

and,

```
eFT> receive -quiet -mode character sourcefile
```

As was true earlier, additional qualifiers and parameters may be passed to "FETCH" (and thus passed through to **RECEIVE**) simply by adding them to the "FETCH" command line when it is invoked.

To remove a previously defined alias, simply define the alias again without a definition:

```
eFT> set alias fetch
```

This will in effect "undefine" the specified alias. Following the command issued above, "FETCH" will no longer be a valid alias.

## NetEx/eFT Aliases vs. Host Aliases

There are two types of aliases, Host Aliases and NetEx/eFT Aliases. Host aliases are either "local aliases" or "remote aliases". That is, their definitions translate to either local host commands or remote host commands. NetEx/eFT aliases translate to NetEx/eFT commands (**SEND**, **RECEIVE**, **CONNECT**, **ASK**, etc.).

In addition to the obvious differences between the two alias types, there are a couple of other distinctions that must be mentioned.

First, NetEx/eFT aliases are defined using **SET ALIAS**. The format is:

| **Command** | **Parameter** |
|---|---|
| `SET ALIAS` | `alias_name ua_command` |

where **ua_command** must translate into a NetEx/eFT command.

Host aliases on the other hand are defined using **SET LOCAL ALIAS** or **SET REMOTE ALIAS**. The format is:

| **Command** | **Parameter** |
|---|---|
| `SET LOCAL ALIAS` | `loc_alias_name host_command` |

and,

| **Command** | **Parameter** |
|---|---|
| `SET REMOTE ALIAS` | `rem_alias_name host_command` |

where **host_command** must be a valid command on the local or remote host respectively.

The other big distinction between NetEx/eFT aliases and host aliases is that NetEx/eFT aliases may have multi-command definitions (see the section entitled "Creating Multi-command NetEx/eFT Aliases"), whereas host aliases can translate only into a single host command. This is not typically a problem since most hosts support command procedures or script files which can then be accessed by a host alias.

The remainder of this section is devoted to issues relating to NetEx/eFT aliases. The topics apply to host aliases as well unless specifically designated as "NetEx/eFT Aliases".

## Creating Multi-command NetEx/eFT Aliases

An alias can translate to a single NetEx/eFT command as shown above where "FETCH" was mapped to **RECEIVE**. NetEx/eFT aliases can also be defined to be a sequence of several commands. The method for defining a multi-command alias is to put each command of the definition on a separate line, where each line, except for the last one, is terminated by the escape character '!'. The escape character tells NetEx/eFT to continue the alias definition on the next line. The following is an example of a three-line "FETCH" alias that connects to "HOSTA", receives a file called "sourcefile", then disconnects:

```
eFT> set alias fetch connect hosta guest netex !
More>>              receive sourcefile !
More>>              disconnect
```

The **SHOW ALIAS** command can be used to display the new "FETCH" alias:

```
eFT> show alias fetch
eFT: FETCH ............... connect hosta guest netex
eFT:                       receive sourcefile
eFT:                       disconnect
```

Notice that NetEx/eFT strips the "!" from the alias definition. It is only needed for the initial definition of the alias.

Although they are very powerful, multi-command aliases are limited to roughly 500 characters in length. It is suggested that if an alias approaches this limit, it should be made into an input script and stored in a file.

Unlike single-command aliases (such as the simplest "FETCH" alias defined earlier), multi-command aliases do not implicitly pass command parameters or qualifiers through to the actual NetEx/eFT commands. NetEx/eFT has no way of determining which command a parameter is destined for unless the alias is set up to pass its own parameters, as explained in the next section.

## Passing Parameters to an Alias

When creating multi-command aliases, it is often desirable to allow parameters to be passed on the alias command line. These parameters can then be substituted into various places within the command sequence in the same way as was done for input scripts.

Refer to the "FETCH" alias defined in the previous section. To make that alias more flexible, one can define it to take as parameters things like userid and password on the **CONNECT** line, and file name on the **RECEIVE** line. The definition then is:

```
eFT> set alias fetch {} connect hosta {1} {2} !
More>>                  receive {3} !
More>>                  disconnect
```

Note the empty "{}" above. These are important when defining alias parameters and will be discussed shortly. The resulting alias becomes:

```
eFT> show alias fetch
eFT: FETCH ............... connect hosta {1} {2}
eFT:                       receive {3}
eFT:                       disconnect
```

To invoke the alias with the same parameters that were embedded in the alias earlier, a user would type:

```
eFT> fetch guest netex sourcefile
```

Parameter 1, represented as "{1}", gets replaced by "guest", parameter 2, represented as "{2}", gets replaced by "netex", and parameter 3, represented as "{3}", gets replaced by "sourcefile". The actual commands that get executed would be equivalent to typing the following:

```
eFT> connect hosta guest netex
eFT> receive sourcefile
eFT> disconnect
```

Now refer to the empty "{}" used in the definition of "FETCH" earlier. This special notation prevents NetEx/eFT from performing parameter substitution while the alias is being defined. The empty "{}" is crucial when defining aliases that use positional parameters or any string substitution (refer to "NetEx/eFT String Substitution" on page 64). If parameter substitution is not turned off NetEx/eFT will attempt to replace the positional parameters with their values at the time the alias is defined (which generally means they get replaced with a null string). The empty "{}" may be placed anywhere within the alias definition and turns off string substitution until the alias definition ends or another "{}" is encountered.

Notice that the empty "{}" appears only on the first line even though the second line is referencing positional parameter 3. The empty "{}" is really a toggle that turns string substitution from off to on (or from on to off) each time it is encountered within a definition. See the section entitled "Disabling String Substitution" on page 91 for further details.

## Accepting Input within a NetEx/eFT Alias

Although aliases that accept parameters are more flexible than those that do not, it may confuse a new user as to which parameters to pass and in what order. Therefore, a more desirable solution is to create aliases that prompt for input. Refer once again to the "FETCH" alias defined in the previous section. It was set up to take three parameters: userid, password, and file name. "FETCH" can be made much more usable by having it prompt for the required parameters, as follows:

```
eFT> set alias fetch {} -
More>>          ask -prompt "User Id?   " uid
More>>          ask -prompt "Password?  " -secure pw !
More>>          ask -prompt "File Name? " fname !
More>>          connect hosta {uid} {pw} !
More>>          receive {fname} !
More>>          disconnect
```

Note the empty "{}" again. These tell NetEx/eFT not to perform string substitution (in this case on variables 'uid', 'pw', and 'fname'), until the alias is executed. The dash (-) on the first line is used to tell NetEx/eFT to continue the definition on the next line.

Quickly breaking this alias apart, the first line simply prompts the user for a userid and reads the response into variable "uid". The next line prompts for a password and reads the result into variable "pw". (The SECURE qualifier is used to tell NetEx/eFT not to echo the response back to the terminal. It should be noted that some versions of NetEx/eFT cannot support this feature due to operating system limitations. For more information, refer to the ASK command in the "Command Descriptions" on page 117.) Next, the file name is requested and read into variable "fname". The **CONNECT** and **RECEIVE** commands are then issued with the appropriate variables to be substituted. Finally, the **DISCONNECT** is performed.

To run the new "FETCH" alias, a user need not know anything about the parameters; the alias will take care of that by prompting for them. Below is an example execution of "FETCH", including user responses:

```
eFT> fetch
User Id?   guest
Password?  _____
File Name? sourcefile
eFT:      < connect results >
eFT:      < receive results >
eFT:      < disconnect results >
```

The output here has been removed to emphasize how prompts can now be used to interact with a user in order to make a more friendly interface. It should be noted here that string functions can also be used within an alias to provide the option of parameter passing or input prompting in the same way as explained in "NetEx/eFT Input Files" on page 92.

## Abbreviating Alias Names

To make life simpler for the user, alias names can be defined to allow abbreviations when they are invoked. For instance, to allow the simplified "FETCH" alias to be invoked by typing only "FET", define it as:

```
eFT> set alias FETch receive
```

Now the definition can be displayed:

```
eFT> show alias fetch
eFT: FETch ....... receive
```

By capitalizing a portion of the alias name (leading consecutive uppercase characters only), a user can define aliases that can be abbreviated or spelled out. In the example above, the "FETCH" alias can now be invoked as "fetch", "fetc", or "fet". By default, aliases are created with all capital letters unless a combination of upper- and lower-case characters are given in its definition, the first character of which must be upper case. The minimum spelling of an alias name includes all letters up to the first lower case letter.

If alias names are defined which cause duplicate abbreviations, (for example, ABc and ABdef), the first alias alphabetically is processed. In the example, AB would cause ABc to execute.

### Defining Multiword Alias Names

For those interested in being even more creative, an alias can be defined with a multiword name. That is, by putting double quotations around the alias name on its definition, the name can contain embedded blanks. For example, to create an alias called "FETCH A FILE", define it as:

```
eFT> set alias "FETch A FIle" receive
```

Its definition can then be displayed as:

```
eFT> show alias "fetch a file"
eFT: FETch A FIle ........ receive
```

And it can be invoked as:

```
eFT> fet a file sourcefile
```

Where "sourcefile" is the file name to receive.

Multiword alias names are particularly useful for users that prefer a more English-like command set. They can also be used to redefine multiword NetEx/eFT commands such as **SHOW HOST**, **SET ALIAS**, etc.

## Debugging a NetEx/eFT Alias or Input Script

It may be necessary from time to time to step through an alias or input script as it is executing, to see exactly what parameters it is using once string substitution has been performed. In the normal case, each command of an alias or input script is silently issued by NetEx/eFT when the alias is invoked. To have NetEx/eFT display each command in its "string substituted" form, set the **VERIFY** qualifier of the **INPUT** command:

```
eFT> set input verify on
```

Now every command issued from the command line gets redisplayed before it is executed, with all positional parameters and string variables replaced by their actual values. This enables a user to debug an alias or input script, to confirm that any substitutions are being performed correctly. Assume the following "FETCH" alias is defined:

```
eFT: FETCH ............. connect hosta {1} {2}
eFT:                    receive {3}
eFT:                    disconnect
```

To debug or verify each command as it gets executed, the **INPUT VERIFY** qualifier is turned on and "FETCH" is invoked with parameters:

```
eFT> set input verify on
eFT> fetch guest "fast netex" sourcefile
eFT: connect hosta guest fast netex
eFT:  ----- connect results -----
eFT: receive sourcefile
```

```
eFT:  ----- receive results -----
eFT: disconnect
eFT:  ----- disconnect results -----
eFT>
```

Another debugging tool is the **CONTINUE** qualifier of the **INPUT** command. By default, NetEx/eFT stops processing an alias or input script as soon as one of its commands fails. For instance, in the example above, if the **CONNECT** failed, the **RECEIVE** and **DISCONNECT** commands would not be processed in the normal case. To tell NetEx/eFT to keep processing an alias or input script even if an error condition occurs, set the **CONTINUE** qualifier of the **INPUT** command:

```
eFT> set input continue on
```

Now when the "FETCH" alias (or any alias) is invoked NetEx/eFT will continue to process the remaining commands even if an error is encountered. The **CONTINUE** qualifier may even be set within an alias definition if the need arises.

The **INPUT** qualifiers **ECHO**, **VERIFY**, and **CONTINUE** are initialized to "off" by NetEx/eFT. Once turned on, they remain on until the user turns them off by typing **SET INPUT VERIFY OFF** or **SET INPUT CONTINUE OFF** respectively. The values of these qualifiers are also determined by the **INPUT** level at any given time. Changing the setting of **ECHO**, **VERIFY**, or **CONTINUE** in an alias or input file changes the setting for the current level and lower levels. Settings for higher input levels are restored when the current input level terminates.

# Error Message Formatting

NetEx/eFT messages consist of the following components:

**SEVERITY** A single character severity level indicator. Possible values are:

| | |
|---|---|
| **I** | Information |
| **W** | Warning |
| **E** | Error |
| **F** | Fatal |

**FACILITY** The facility or subsystem name generating the message. This will generally be some version of the following:

| | |
|---|---|
| **UA** | NetEx/eFT host independent message. |
| **UAxxx** | NetEx/eFT host dependent message where xxx represents the product number of the host generating the error (e.g. EFT263, EFT213, etc.) |
| **SI** | NetEx/eFT Service Initiator (SI) host-independent message. |
| **SIxxx** | NetEx/eFT Service Initiator (SI) host-dependent message. |
| **MUX** | NetEx/eFT Multiplex Server (MUX) host-independent message. |
| **MUXxxx** | NetEx/eFT Multiplex Server (MUX) host-dependent message. |
| **NETEX** | A NETEX generated error. |
| **opsys** | An operating system specific error where OpSys is replaced by the operating system name generating the error. |

**CODE** The unique error or message code.

**TEXT** The single line message text describing the error code.

The format of a message display is controlled by the **OUTPUT** command qualifier **FORMAT**. Specific components of a NetEx/eFT message are extracted using the string function **msg()** (described in the section on "String Functions"). The default message format can be displayed as:

```
eFT> show output format
eFT: FORmat ....... {msg("text")} ({msg("facility")}-{msg("code")}).
```

With this format defined, a simple “Invalid command” error would generate the following:

```
eFT> badcommand
eFT: Invalid command 'badcommand' (UA-4708).
```

The user can modify the message format simply by changing the value of the **FORMAT** qualifier of the **OUTPUT** command. The value can be any string so long as it includes some reference to string substitution when it gets interpreted. That is when **OUTPUT FORMAT** is defined, it must disable string substitution using the “{}” syntax. An invalid **FORMAT** specification will result in NetEx/eFT returning the value to its original, default value. This is done to make sure error messages are properly displayed even if they were inadvertently shut off.

The following example modifies the error message format to print only the error severity, facility, and code. Note the use of {}

```
eFT> set output format {} {msg("severity")}:{msg("facility")}-
{msg("code")}
```

With this message format, the same “Invalid command” error would generate the following display:

```
eFT> badcommand
eFT: W:UA-4708
```

It is advised that error message format tailoring should be left up to the site administrator. Most users will never need to modify the default format.

# NetEx/eFT Code Conversion

NETEX is ordinarily responsible for performing code conversion between computer systems. NetEx/eFT provides an alternative code conversion facility intended for environments requiring more flexibility than that offered within NETEX. NetEx/eFT code conversion adds the following capabilities:

- It supports ASCII to ASCII and EBCDIC to EBCDIC code conversion, allowing a site to handle differences among “like” conversion tables (Unisys A versus IBM EBCDIC, for example).
- It supports full (256-character) ASCII as well as the NETEX 128-character ASCII tables. This is particularly useful for handling the variety of country codes that appear in the last half of the ASCII tables.
- It allows a site to specify incoming code conversion and outgoing conversions separately.
- It allows NetEx/eFT to offer optional data verification facilities (CRC) for character file transfer as well as bit-stream transfer. Refer to “NetEx/eFT Data Verification” on page 106 for more information.

All NetEx/eFT code conversion is controlled by means of the **TRANSLATE** command. Refer to the **TRANSLATE** command as described in “Command Descriptions” on page 117. Using **TRANSLATE**, a site administrator can define code conversion tables, review the current tables, and enable or disable the NetEx/eFT code conversion facility.

Although a user may use the **TRANSLATE** command to specify changes to the conversion tables “on the fly,” it is strongly suggested that code conversion be treated as a site operations issue and that any code conversion table changes be established at a site level by means of the **TRANSLATE SEARCH** qualifier. By using the **TRANSLATE SEARCH** path mechanism, a site administrator can define the tables and enable NetEx/eFT code conversion as part of the **CONNECT**/**LOGIN** process to a remote system.

In NetEx/IP environments, when NetEx/eFT code conversion is enabled, it replaces NetEx code conversion in all communications between systems. To protect the NetEx/eFT protocol from code conversion changes the following characters may not be modified:

Uppercase alphabetic characters (A-Z)
Digits (0-9)
Space, equal sign (=), and null

If NetEx/eFT code conversion is to be used for only certain file transfers, it is recommended that aliases be set up to control enabling and disabling of the facility.

# NetEx/eFT Data Verification

NetEx/eFT offers an optional data verification facility or Cyclic Redundancy Check (CRC) that can be involved on file transfers. When CRC is turned on as a **SEND** or **RECEIVE** qualifier, NetEx/eFT appends a block number, and the result of a CRC calculation to each block of the file transferred. If a block is lost or if the source CRC calculation does not match the destination calculation, the file transfer is aborted.

A sample alias, "SENDCRC", shown below, will attempt to resend an aborted file a specified number of times. To make this alias easier to read, it uses the **ASSIGN** and **TEST** aliases that are defined in NetEx/eFT.

```
* SENDCRC - send a file with CRC enabled - if the transfer
*           fails with a retryable error (this includes a
*           CRC failure), sleep for a while and try again.
*           A retry counter limits the number of attempts.
*
set alias sendcrc {} on error goto check!
send: send -crc {0}!
exit success!
check: test msg("retry") eqs "N" exit error!
{sleep(10)}!
assign count inc count!
test count le 100 goto send!
text Maximum retries exceeded - SENDCRC cancelled.!
exit error
```

Since during normal operation a CRC error should be extremely rare, the "SENDCRC" alias (or similar logic incorporated in user-defined scripts) is an effective way to guarantee the delivery of data.

If CRC is performed on character files, NetEx/eFT code conversion is used (refer to "NetEx/eFT Code Conversion" on page 105) automatically. Although the CRC facility invokes the code conversion facility without any user action required, it must be noted that the code conversion facility will process the data based on the current user conversion tables (including **SEARCH** paths) only if translation has been enabled (i.e., **TRANSLATE ON**). If translation has not been enabled, NetEx/eFT uses its own default tables and does not use any user specified translation tables or search paths.

The CRC algorithm is performed using 32-bit values. The integrity of a data stream is checked by comparing its state at the sending host and the receiving host. Each character in the data stream is used to generate a value based on a polynomial. The values for each character are then added together. This operation is performed at both ends of the data transmission, and the two results compared. If the results are different, an error has occurred during transmission.

# IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER),

# and (NONE)

This section assumes the reader has a working knowledge of the NetEx/eFT commands as well as a general understanding of the **SEARCH** qualifier for each of them.

The NetEx/eFT commands **CONNECT**, **INPUT**, **HELP**, and **TRANSLATE** have **SEARCH** qualifiers associated with them. The **SEARCH** qualifiers generally instruct NetEx/eFT to look for a file or set of files (depending on the command it is associated with), with the given name and in the specified location. One option the user always has is to give the entire file specification (path and file name) for each path or file that is to be searched. The other option, implemented simply as a shortcut for the user, is to specify the keywords **(SITE)**, **(USER)**, or **(NONE)**, in some form, as the definition for qualifier **SEARCH**. The **SEARCH** keywords have the following "generic" definitions:

**(SITE)** This keyword refers to the NetEx/eFT site or root directory on the local or remote host. On a z/OS host, **(SITE)** is a partitioned data set. **(SITE)** is where system wide NetEx/eFT files (members) are stored. The actual value for **(SITE)** can be determined by displaying the "hidden" **LOCAL** or **REMOTE** qualifier called **ROOTDIR**:

```
eFT> show local rootdir
```

or

```
eFT> show remote rootdir
```

**(USER)** This keyword refers to the user's login or home directory (or equivalent) on the local or remote host. On a z/OS host, **(USER)** refers to the user's login TSO/E prefix. Often **(USER)** is where users keep personal files (such as NetEx/eFT startup files) that are intended for tailoring NetEx/eFT to personal taste. The actual value for **(USER)** can be determined by displaying the local or remote **HOMEdir** qualifier.

**(NONE)** This keyword simply tells NetEx/eFT to not search for any files. It is different than leaving a **SEARCH** qualifier value blank. A **SEARCH** qualifier with no definition often implies some default files should be located. Specifying **(NONE)** as a definition for **SEARCH** specifically tells NetEx/eFT not to look for any files.

The keywords **(SITE)** and **(USER)** can be used as they appear above, or with a member or file name appended to them respectively. Appending a member or file name to **(SITE)** or **(USER)** tells NetEx/eFT to look in the specified location for the given member or file name. When specified without any appended text, the keywords may have implied names that NetEx/eFT attempts to locate.

The rest of this section addresses each NetEx/eFT command that has a **SEARCH** qualifier and how that qualifier interprets the keywords **(SITE)**, **(USER)**, and **(NONE)**.

**CONNECT** The **SEARCH** qualifier for the **CONNECT** command is used to locate NetEx/eFT startup files on the remote host following a successful connection. The default definition for **SEARCH** is the string "(SITE) (USER)". When the connection is being made to a z/OS host, **(SITE)**, when declared without a member name, always refers to the member "SSERVER" on the remote z/OS host in the **(SITE)** partitioned data set. When **(USER)** is declared without a file (or data set) name, it implies a sequential data set by the name of "SERVER.UA" appended to the user's TSO prefix. **(NONE)** tells NetEx/eFT to not process any remote startup files.

For example, to define the **CONNECT SEARCH** qualifier to locate the default site startup file, the default user startup file, and a third startup file called "NEW" located in the NetEx/eFT **(SITE)** partitioned data set, issue the following command:

```
eFT> set connect search (SITE) (USER) (SITE)NEW
```

The user should be reminded that the order here is important. NetEx/eFT uses the **SEARCH** definition from left to right. In the example above, "(SITE)" would be searched first, then "(USER)", and finally "(SITE)NEW".

**HELP**

The **HELP SEARCH** qualifier is used to locate NetEx/eFT help files on the local host. The default definition for SEARCH is "(SITE)". **(SITE)**, when declared without a member name, always implies the member "EFTHELP" in the **(SITE)** partitioned data set. **(USER)** has the default file name "HELP.UA" associated with it. Users may define **SEARCH** as a string containing "(SITE)xxx" or "(USER)xxx" where "xxx" is the name of a user defined help file, either a member name or sequential data set name, respectively.

For example, issue the following command to define the HELP SEARCH qualifier to look for a user created help file called "MYHELP.HLP.UA" appended to the user's TSO prefix, and the default NetEx/eFT help file:

```
eFT> set help search (USER)myhelp.hlp (SITE)
```

**INPUT**

The **SEARCH** qualifier for the **INPUT** command (explicit or implicit) is used to locate NetEx/eFT input scripts. There are no default file names associated with **(SITE)** or **(USER)** for the **INPUT** command **SEARCH** qualifier. This means that specifying **(SITE)** or **(USER)** without a suffix is the same as specifying **(NONE)**. You can suffix **(SITE)** or **(USER)** with any character that will generate a valid z/OS data set name. Including an asterisk (*) in a suffix causes the input file name to be substituted at that point.

**Note:** Do not specify "(SITE)*" as it will find the distributed help text files and attempt to execute them as the alias commands.

The search order for an explicit **INPUT** command is to first check the user's directory (TSO/E prefix) for a data set with the input file name, and then check the **INPUT SEARCH** qualifier path(s). The search order for an implicit **INPUT** command (generated just before the "Invalid Command" error) skips the user's directory search and proceeds directly with the **INPUT SEARCH** qualifier path(s).

For example, the **INPUT SEARCH** qualifier can be defined to search all the members in the site partitioned data set that end in "UA" by issuing the command:

```
eFT> set input search (SITE)*UA
```

Now any input requests (via the **INPUT** command or implied), will only require the member name minus the 'UA'. For example, if a NetEx/eFT input script by the name of "MYJOBUA" is in the root directory, it can be invoked as:

```
eFT> input myjob
```

or

```
eFT> myjob
```

Note that if a sequential data set with the name "TSO.PREFIX.MYJOBUA" exists, the **INPUT** command in the first example will process that data set instead, whereas the "myjob" command will not.

**TRANSLATE**

The **SEARCH** qualifier for the **TRANSLATE** command is used to locate user defined script files. The default definition for **SEARCH** is "(SITE)". **(SITE)**, when declared without a member name, always refers to a member in the site partitioned data set on the local host with the name "{HOSTCODE:REMOTE}", where "{HOSTCODE:REMOTE}" is the string substitution syntax for the host character code (ASCII8, EBCDIC, etc.) of the remote host. **(USER)** has no implied file name associated with it for the **TRANSLATE** command. Users may define **SEARCH** as a string containing "(SITE)xxx" or

"(USER)xxx" where "xxx" is the name of a user defined script file (member name or sequential data set, respectively) containing **TRANSLATE** commands.

For example, issue the following command to define the **SEARCH** qualifier for the **TRANSLATE** command to locate a user created script file called "IBMTODEC.UA" located in the user's TSO prefix:

```
eFT> set translate search (USER)IBMTODEC
```

# User-Definable HELP Files Under IBM z/OS

A site may create its own **HELP** files that NetEx/eFT will look for upon request. User-definable help files allow a site or user to write help text for newly created aliases and input scripts. The **HELP** qualifier **SEARCH** is used to tell NetEx/eFT to look for additional help files. By default, the **HELP SEARCH** qualifier is defined as:

```
eFT> show help search
eFT: SEArch ............ (SITE)
```

**(SITE)** is a special **SEARCH** keyword used by z/OS NetEx/eFT to indicate the default help file member "EFTHELP" from the local NetEx/eFT site partitioned data set directory. This help file contains all NetEx/eFT commands, qualifiers, examples, etc. To instruct NetEx/eFT to also look in a site or user defined help file (partitioned data set member or sequential data set), for example, "JONESX.MYALIAS.UA", type the following:

```
eFT> set help search 'JONESX.MYALIAS.UA' (SITE)
```

The order of the **SEARCH** list is important. In the example above, NetEx/eFT will look first in the "MYALIAS.UA" help file upon any help request. If no match is found there, NetEx/eFT reads the next help file in the SEARCH list, in this case the **(SITE)** default help file.

It may be desirable to store any site help files along with the NetEx/eFT default one, in the root directory. The **(SITE)** keyword allows for appending member names as shown below:

```
eFT> set help search (SITE)myalias (SITE)
```

This example instructs NetEx/eFT to first look for any help requests in the member "MYALIAS" in the **(SITE)** partitioned data set, followed by the NetEx/eFT default file "EFTHELP" from the same PDS. Refer to "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106 for more information.

To create a site or user help file, use any standard text editor available on the system. The format of the help file must follow the example below:

```
Topic_Level  Topic_Name
.....................................
...............Help text.............
.....................................
Topic_Level  Topic_Name
.....................................
...............Help text.............
.....................................
```

Where:

**Topic Level** is a numeric value from 1 to 9 indicating the help level representing this "Topic_Name". The first "Topic_Level" of all help files must be level 1. Subsequent topics can then be sub-topics (2, 3, etc.) to the level 1 topic, or new top-level topics.

**Topic Name** is the character string representing the help topic. This is the name the user types to obtain help text (e.g. "HELP SENd"). This string should be limited to 15 characters in length for output formatting purposes. The "Topic_Name" may include upper-case characters followed by lower-case characters in order to allow an abbreviation on the help request.

**Help Text** is the actual help text the user will see in response to a help request. All help text should be character (text) data only. Users should be careful to not include unprintable characters, including tabs (multiple spaces are recommended) and control characters.

The following is an abbreviated version of the NetEx/eFT standard help file to be used as a reference when creating site or user help files:

```
1. ASK
      The ASK command prompts a user for one or more responses...
2. QUAlifiers
      This is where the qualifiers for the ASK command would be described
      within the help file -- as a sub-topic to the ASK command.
2. EXamples
      And this is where any ASK examples would be shown.  This sub-topic
      to ASK is the same level as QUAlifiers.
1. CONnect
      The CONNECT command is used to establish a connection to a remote
      host on the network...
      (This is a new top-level topic.)
2. EXamples
      Any examples for the CONNECT command would appear here as a sub-
      topic.
3. MORE_Examples
      This is here just to show where a level three help sub-topic would
      appear.
1. DISconnect
      And so on...
```

Note that under a top-level topic there may be multiple sub-topics and even sub-topics to them. It is up to the site to make sure these user written help files are formatted properly. It may be useful to refer to the standard NetEx/eFT help file as a guide.

# Running an IBM z/OS NetEx/eFT Client in Batch

The IBM z/OS NetEx/eFT Client can also be executed in a z/OS batch (background) environment as either a z/OS program or under control of TSO/E. Running as a z/OS program does not support local (TSO/E) command execution. Running the Client in batch is typically done for large file transfers, tape file transfers, or for time dependent batch jobs.

For example, a company may want to perform nightly backups of three smaller departmental HPE OpenVMS systems. The OpenVMS systems, however, do not have 24 hour a day attended operations. Sometime after midnight, the z/OS job scheduling package can submit three z/OS batch jobs, each executing an IBM/z/OS NetEx/eFT Client program. The input text in the z/OS JCL directs NetEx/eFT to connect to an OpenVMS host, initiate the OpenVMS "BACKUP" utility, transfer the OpenVMS backup output to a z/OS data set (disk, tape, etc.), and disconnect. Upon successful transfer of the backup data, the Client step ends with a condition code of zero. If the backup data transfer was unsuccessful, a non-zero condition code is set. Either way the z/OS job scheduling package processes the condition code, and if it is nonzero, automatically notifies the z/OS operations personnel that corrective action is needed.

There are two methods to execute the Client in batch:

- Running the Client as a z/OS program. Local command execution is not supported.
- Running the Client in batch under control of TSO/E. Local command execution is supported, with the normal TSO/E background (batch) limitations.

The installation may use either method as needed.

# Tailoring the Client Execution JCL/CLISTs

The instructions below are provided as examples of how to set up z/OS JCL for the NetEx/eFT client. Your installation standards and procedures may substantially change these examples in which case you may need to contact your site administrator to provide further assistance.

If you are going to run the client as a z/OS program without running under the control of TSO/E (no local command execution), tailor the example JCL in Figure 3 on page 111 according to the following instructions.

Change the following to meet your installation requirements.

- The JOB card(s)
- The data set name "DSN=EFT.N24.LOAD" on the STEPLIB card.

  Change this name to the data set name at your installation that contains the client program module. This STEPLIB card can be deleted if your installation made the client program available via your z/OS link list concatenation.
- You may also specify any valid client qualifiers needed at your installation in the "PARM=" text. The example JCL sets a global variable memory size of 4096 bytes. Refer to "Invoking NetEx/eFT on IBM z/OS" on page 9 for more information on qualifiers.
- CLIST allocations of STDIN should be changed to include a block size equal to or greater than the screen size of the 3270 model that you are using. For example:

  ```
  ALLOC F(STDIN)    DS(*) REUSE BLKSIZE(4096) RECFM(U)
  ```
- Change the NetEx/eFT commands following the STDIN card to meet your processing requirements. The example shows three commands, **CONNECT**, **SEND**, and **DISCONNECT**.

Submit the job for execution to run the Client.

```
//EFTCJCL  JOB ,'EFT CLIENT',CLASS=A,MSGCLASS=X
//CLIENT   EXEC PGM=NUACLIEN,REGION=2048K,PARM='-GLOBAL 4096'
//STEPLIB  DD DSN=EFT.N24.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//STDIN    DD *
CONNECT host userid password
SEND mvs.dataset.name
DISCONNECT
/*
```

**Figure 3. Client example JCL**

If you are going to run the Client in batch under the control of TSO/E (allowing local command execution), tailor the example JCL in Figure 4 on page 112 according to the following instructions.

Change the following to meet your installation requirements.

- The JOB card(s)

- The data set name “DSN=SYS1.CMDPROC” on the “SYSPROC” card.

  Change this name to the data set name at your installation that contains the client TSO/E CLIST “EFTUSER”.

- The TSO/E “PROFILE PREFIX(EFT)” command. This can be used to set the initial TSO/E prefix and NetEx/eFT “Directory” qualifier for user startup file processing. The user startup file that would be processed using the example prefix is “EFT.CLIENT.UA” (if it existed). If user startup file processing is not needed, this command can be deleted.

- You may also specify any other valid client qualifiers needed at your installation following the “EFTUSER” command. The example JCL sets a global variable memory size of 4096 bytes. Refer to “Invoking NetEx/eFT on IBM z/OS” on page 9 for more information on qualifiers.

- Change the NetEx/eFT commands following the STDIN card to meet your processing requirements. The example shows three commands, **CONNECT**, **SEND**, and **DISCONNECT.**

Submit the job for execution to run the Client.

```
//EFTCJCL  JOB ,'EFT CLIENT',CLASS=A,MSGCLASS=X
//SERVER   EXEC PGM=IKJEFT01,REGION=2048K
//SYSPROC  DD DSN=SYS1.CMDPROC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
PROFILE PREFIX(EFT)
%EFTUSER SCR('-GLOBAL 4096')
//STDIN    DD *
CONNECT host userid password
SEND mvs.dataset.name
DISCONNECT
/*
```

**Figure 4. Client under TSO/E example JCL**

## Running an IBM z/OS Standalone NetEx/eFT Server

The Standalone Server can be used to allow a remote host access to z/OS without establishing a TSO/E logon session. The Standalone Server can be used in place of the Multiplex Server or can execute concurrently with the Multiplex Server but will only remain active for one remote user session. When the remote user disconnects from the Standalone Server, the Standalone Server will terminate. It is typically used for large file transfers, tape file transfers, or for data dependent batch jobs.

For example, an application may exist on an HPE OpenVMS system that processes a division’s sales data each night. At some point after midnight, the OpenVMS application needs to send a file to the corporate accounts payable system on the z/OS host. The z/OS job scheduling package submits a two-step job at midnight, the first step being a Standalone Server, and the second step being an accounts payable database update. The Standalone Server step waits until the OpenVMS application executes OpenVMS NetEx/eFT and connects to the z/OS system with the correct **-SERVICE** name. After the file transfer completes successfully and the OpenVMS system disconnects, the Standalone Server step ends with a condition code of zero. The next job step now executes to update the accounts payable database using the transferred file.

If the z/OS accounts payable application could only be run before 3:00 A.M., the **-TIMEOUT** qualifier could be set to terminate the Standalone Server automatically. If the Standalone Server time limit expires or an error is detected during file transfer, a nonzero condition code is set causing the z/OS job scheduling package to be notified and the accounts payable update to be skipped. The OpenVMS application would have to save the data

until the next night or the operations staff could reschedule the OpenVMS and z/OS applications according to their restart procedures.

There are two methods to execute the Standalone Server in batch:

- Running the Standalone Server as a z/OS program does not interface with ACF/VTAM or TSO/E allowing remote users to connect to IBM/z/OS NetEx/eFT without ACF/VTAM or TSO/E being active, however, remote command execution is not supported. **The sever must be configured using the 'PARM' parameter on the 'EXEC' statement. This has a 100-character maximum limitation. This is a z/OS restriction. Configuring this server to run in secure mode is very problematic.**
- Running the Standalone Server in batch under control of TSO/E does not interface with ACF/VTAM allowing remote users to connect to IBM/z/OS NetEx/eFT without ACF/VTAM being active or using ACF/VTAM resources. **Remote command execution is supported, but TSO/E command output is not returned to the remote user. The server is configured as a TSO command. This removes the 100-character limitation imposed by the operating system.**

The installation may use either method as needed.

The NetEx/eFT Stand-Alone Server (or Responder) can be invoked from a TSO/E command line (inter-actively or in batch), as:

```
NUASSERV [-keyword value]
```

Where:

**NUASSERV** is the program name of the NetEx/eFT Standalone Server.

**-keyword value** (optional) specifies optional command line keywords that may be given to affect the operation of the NetEx/eFT session. The following are valid keywords:

**-BLOCKsize** specifies an alternative default NetEx blocksize value in which to offer. The default is 16384. The **USER_BLOCKSIZE** environment variable can also be used to set this value.

**--CERTDB** For secure connections, this specifies the name the certificate database to use. For a GSKKYMAN database, this will be the dataset. The **CERTPW** MUST specify the read access password. For a RACF database, this will be the userid/keyring combination given to you by your RACF administrator. The **CERTPW** must NOT be coded.

**--CERTLB** For secure connections, this specifies the name (label) of the certificate to use.

**--CERTPW** For secure connections, this specifies the read password to the GSKKYMAN database. This must NOT be coded if you are using RACF.

**--FIPSLVL** For secure connections, this specifies the level of FIPS encryption that is required.

**0** A FIPS formatted certificate database is not required.
**1** The certificate database must be in FIPS format.
**2** The certificate must meet FIPS Level 2 requirements.
**3** The certificate must meet FIPS Level 3 requirements.

See the documentation for the "gsk_fips_state_set()" routine for the complete requirements for each level.

**--HOMEdir** specifies the name of the user's "login" or "home" directory when NetEx/eFT is invoked. Changing this keyword's value redefines the location NetEx/eFT uses

to locate user startup files. The **USER_HOME** environment variable can also be used to set this value.

**-OUTput** specifies the name of an output file that is to receive the server output from this session. The default is the terminal or the batch log file. **The USER_OUTPUT** environment variable can also be used to set this value.

**-PASSword** specifies an optional password that will be checked (by the server) after a connection is established to this server. A connecting initiator must pass a password matching in length and exact character case. The **USER_PASSWORD** environment variable can also be used to set this value.

**-PREfix** an alternative server prefix string that precedes all server displays. The default is "Server: ".

**-ROOTdir** specifies the name of the installed NetEx/eFT root directory containing site specific server help and startup files. There is generally no reason to modify this keyword. The **USER_ROOT** environment variable can also be used to set this value.

**-SEArch** specifies the search path NetEx/eFT follows to locate server startup files following a successful connection. This value is only used if the **CONNECT SEARCH** path from the initiator is empty. The default is "(SITE) (USER)". The **USER_SEARCH** environment variable can also be used to set this value.

**-SECure** specifies if a secure connection is to be established.

**-SERvice** specifies a service name to offer. The default is "EFT". The **USER_SERVICE** environment variable can also be used to set this value.

It is recommended that the service name you choose for a Standalone Server not match the service name chosen for the Multiplex Server or other Standalone Servers. If any service names are the same, a remote user may be connected to any server offering the same name, causing confusing results.

**-SSLCIPHERS** specifies which ciphers to use. It is a list of 4-digit numbers. If omitted, all ciphers are available. See the IBM document "z/OS Cryptographic Services System SSL Programming" for the four-digit codes

**-SSLPROTOCOL** specifies which protocols to use. ALL (default) allows all protocols to be used. "ALL:-tlsv1" indicates all protocols except tlsv1 may be used. The valid protocols are: ALL, TLSV1, TLSv1.1 TLSv1.2. Addition may also be used, for example: "TLSv1.1+TLSv1.2".

**-TIMEOUT** specifies a value in seconds for which the standalone server will offer the service. The standalone server exits with a nonzero condition code after the specified time expires. The **USER_TIMEOUT** environment variable can also be used to set this value.

## Tailoring the Standalone Server execution JCL

The instructions below are provided as examples of how to set up z/OS JCL for NetEx/eFT. Your installation standards and procedures may substantially change these examples in which case you may need to contact your site administrator to provide further assistance.

Two JCL examples are provided; one for running the Standalone Server as a z/OS batch program, and one for running the Standalone Server under the control of TSO/E. One or both may be used depending on your installation's requirements.

If you are going to run the Standalone Server as a z/OS program that is not running under the control of TSO/E (no remote command execution), tailor the example JCL in Figure 5 on page 115 according to the following instructions.

Change the following to meet your installation requirements.

- The JOB card(s)
- The data set name "DSN=EFT.N24.LOAD" on the STEPLIB card.

  Change this name to the data set at your installation that contains the Standalone Server program module. This STEPLIB card can be deleted if your installation made the Standalone Server available via your z/OS link list concatenation.
- The service name "-SERVICE EFTSA".

  It is recommended that the service name you choose for a Standalone Server not match the service name chosen for the Multiplex Server or other Standalone Servers. If any service names are the same, a remote user may be connected to any server offering the same name, causing confusing results.
- The "-SECURE" parameter.

  If coded, a secure connection will be established. This requires the –CERTDB, -CERTLB, -CERTPW (if a GSKKYMAN database is being used) and -FIPSLVL (if a FIPS-compliant database is used) be coded.

Submit the job for execution to run the Standalone Server.

```
//EFTSJCL  JOB ,'EFT SA SERV',CLASS=A,MSGCLASS=X
//SERVER   EXEC PGM=NUASSERV,REGION=2048K,
//*           PARM data has a 100 character maximum imposed by z/OS
//            PARM='-SERVICE EFTSA -TIMEOUT 600'
//STEPLIB  DD DSN=EFT.N24.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

**Figure 5. Standalone Server example JCL**

If you are going to run the Standalone Server as a batch job under the control of TSO/E (allowing remote command execution without returned output), tailor the example JCL in Figure 6 on page 116 according to the following instructions.

Change the following to meet your installation requirements.

- The JOB card(s)
- The data set name "DSN=EFT.N24.LOAD" on the STEPLIB card.

  Change this name to the data set at your installation that contains the Standalone Server program module. This STEPLIB card can be deleted if your installation made the Standalone Server available via your z/OS link list concatenation.
- The TSO/E "PROFILE PREFIX(EFT)" command. This can be used to set the initial TSO/E prefix and NetEx/eFT **DIRectory** qualifier for user start up file processing. The user startup file that would be processed using the example prefix is "NUA.SERVER.UA" (if it existed). If user startup file processing is not needed, this command can be deleted.

- The service name "-SERVICE EFTSA"

  It is recommended that the service name you choose for a Standalone Server not match the service name chosen for the Multiplex Server or other Standalone Servers. If any service names are the same, may connect a remote user to any server offering the same name, causing confusing results.

- You may also specify any other valid Standalone Server qualifiers needed at your installation following the NUASSERV command. The example JCL shows a 10-minute time limit, "-TIMEOUT 600", being set.

Submit the job for execution to run a non-secured Standalone Server.

```
//EFTSJCL  JOB ,'EFT SA SERV',CLASS=A,MSGCLASS=X
//SERVER   EXEC PGM=IKJEFT01,REGION=2048K
//STEPLIB  DD DSN=EFT.N24.LOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
PROFILE PREFIX(EFT)
NUASSERV -SERVICE EFTSA -TIMEOUT 600
```

**Figure 6. Non-Secure Standalone Server under TSO/E example JCL**

Submit the job for execution to run a secured Standalone Server under TSO/E.

```
//EFTSJCL  JOB ,'EFT SA SERV',CLASS=A,MSGCLASS=X
//SERVER   EXEC PGM=IKJEFT01,REGION=2048K
//STEPLIB  DD DSN=EFT.N24.LOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
PROFILE PREFIX(EFT)
NUASSERV -SERVICE EFTSA -TIMEOUT 600 +
   -CERTDB /users/huhned/test.kdb -CERTPW testtest +
   -CERTLB zos5r -SECURE
```

**Figure 7. Secure Standalone Server under TSO/E example JCL**

# Command Descriptions

This section contains descriptions of these commands:

ASK
CONNECT
CONTINUE
DISCONNECT
EXIT
GOTO
HELP
INPUT
LOCAL
ON
OUTPUT
QUIT
RECEIVE
REMOTE
SEND
SET
SET ALIAS
SET GLOBAL
SET HOST
SET VARIABLE
SHOW
SHOW ALIAS
SHOW GLOBAL
SHOW HOST
SHOW QUALIFIER
SHOW VARIABLE
TEXT
TRANSLATE

The command descriptions or qualifiers for some commands may differ slightly between hosts. These variations are detailed in the software reference manual for that host.

# ASK Command

## Description

The **ASK** command prompts a user for one or more responses. If multiple responses are desired, the command line should contain multiple variables to receive the user's input. For example, if you are prompting a user for name and number, you should declare two variables (e.g., uname, unum) on the **ASK** command line in order to save both responses. If a user enters a carriage return (or <ENTER>), the variable gets defined to a null string unless a default value is supplied with the **DEFAULT** qualifier.

**ASK** variables get set to whatever the user types as input, whether it be one word or an entire string. If multiple variables are declared, the first one is set to the first word of input, the second one gets set to the second word of input, and so on. The last variable declared gets defined to the remainder of the input string. If you prompt for more input than the user gives (e.g., you declare four variables and the user types just two words of input), the remaining variables are defined to be a null string.

Variable names specified on the **ASK** command line must be alphanumeric and no longer than 20-characters in length. If no variable names are specified at all on the command line, **ASK** still prompts the user, but no variables get defined. (This can be used to pause during input processing).

In addition to the **ASK** command, you can define a variable by typing "*SET VARIABLE name value*". You can display the list of all session variables by typing **SHOW VARIABLES**.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| ASK | [-DEFault string]<br>[-PROMpt string]<br>[-SECure  \| ON  \| ]<br>\| OFF \|<br>[-TIMEout seconds] | [VAR1 [VAR2...]] |

Where:

**ASK** (required) the verb for this command.

**-DEFault** (optional) the default string passed to **ASK** if the user does not provide one. This default string gets processed as if the user had typed it. The minimum spelling is -DEF.

**-PROMpt** (optional) a string used for the **ASK** prompt. You need to enclose the **PROMPT** string in double quotes in order to include trailing spaces on the prompt or to use multiple word prompts. The minimum spelling is -PROM.

**-SECure** (optional) tells NetEx/eFT not to echo the user's response to this **ASK** command. **SECURE** is used primarily for reading in a user's password or any other time security is a concern. This value should be set to either ON or OFF. The default is OFF unless **SECURE** is specified. The minimum spelling is -SEC.

**-TIMEout** (optional) tells NetEx/eFT how long to wait, in seconds, for a response. This integer value is between 0 and 10000. This parameter is NOT functional in z/OS. The default value is 0. The minimum spelling is –TIME.

**variables** (optional) zero or more variable names separated by a space that will receive the user's response to the **ASK** command.

## Examples

To prompt a user for input into variable “name” with a default name of “Ed”, and a prompt of “Name?”, type:

```
eFT> ask -prompt "Name? " -default Ed name
Name? Joe Smith
```

The user’s response “Joe Smith” is read into variable name. Now using standard NetEx/eFT string substitution syntax, you can display the value of name with the **TEXT** command:

```
eFT> text Hello {name}.
eFT: Hello Joe Smith.
```

Alternatively, you can display the value of variable name: as:

```
eFT> show variable name
eFT: NAME ............. Joe Smith
```

If you wanted to prompt for the variables “day” and “date”, you could type:

```
eFT> ask -prompt "Enter day and date: "day date
Enter day and date: Thursday June 6, 2019
```

The value of variable “day” would become “Thursday” and the value of variable “date” would be the remainder of the line which is the string “June 6, 2019”.

```
eFT> show variable
11:58:53 MVS:
11:58:53 MVS:  DATE .............. June 6, 2019
11:58:53 MVS:  DAY ............... Thursday
11:58:53 MVS:  NAME .............. Joe Smith
11:58:53 MVS:
```

## Related Topics

INput Command
SHOW Command
TEXt Command

# CONnect Command

## Description

The **CONnect** command is used to establish a connection to a remote host on the network. The host name specified must exist in the local network configuration. By default, **CONnect** attempts to connect to the service named **EFT** offered on the remote host. If this service is not offered the connection request will eventually time out. A remote Service Initiator or Multiplex Server is usually the one making the service offering. You can set or display the service name with the **SET CONnect SERvice** and **SHOW CONnect SERvice** commands respectively. The recommended way for connecting to a remote host is to use the **LOGIN** or **LOGON** alias. These aliases prompt the user for a hostname, username, and password, (and account and profile for **LOGON**) then issues the appropriate **CONnect** command for the user.

The connect process logs you in to the remote host using its standard login procedure; valid usernames and passwords must be provided to accomplish this. Some systems provide a default NetEx/eFT login username and password. If the system you are connecting to supports this feature, you could optionally leave off these two parameters.

When a connection is first established it becomes the current "active" connection -- any **REMOTE** command refers to the host associated with that connection. When a connection to another host is attempted, the old "active" connection is put into an idle state. NetEx/eFT allows as many as ten connections to exist at a time, although one or two connections is normally all that a user would have use for. You can switch from one connection to another by means of the **SET HOST** command.

Connecting to certain hosts often takes more than a few seconds. Therefore, NetEx/eFT displays intermediate **CONnect** messages informing you of its current connect status. If a connection cannot immediately be established due to the service not being offered on the specified host or because the service is busy, **CONnect** will retry every **INTerval** seconds for a total of **TIMeout** seconds (**INTerval** and **TIMeout** are **CONnect** command qualifiers). Intermediate connect messages display both success and failure information.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| CONnect | [-ACCount code]<br>[-ADAPTer string]<br>[-APPlication string]<br>[-BLOCKsize bytes]<br>[-COMmand name]<br>[-INTerval seconds]<br>[-PASSword pw]<br>[-PROFile name]<br>[-PROJect code]<br>[-QUIet \| ON \|]<br>\| OFF \|<br>[-SCRIpt filename]<br>[-SEArch string]<br>[-SECondary pw]<br>[-SECUre]<br>[-SERvice name]<br>[-SSLCIPHERS nnnn]<br>[-SSLPROTOCOL types<br>[-SITE string]<br>[-TIMeout seconds]<br>[-USERname user]<br>[-VERBose \| ON \|]<br>\| OFF \| | host [user] [password] [arg ...] |

Where:

**CONnect** (required) the verb for this command. The minimum spelling is CON.

**-ACCount** (optional) login account code that may be used by the host in which you are attempting to connect. The minimum spelling is -ACC.

**-ADAPTer** (optional) specify an alternate NetEx/IP (H210IPz) subsystem name. The minimum spelling is -ADAPT.

**-APPlication** (optional) identify the login application on the remote host. The minimum spelling is -APP.

**-BLOCKsize** (optional) local host's maximum network block size in bytes. This size gets sent to the remote host on a **CONnect** and a new negotiated NetEx/IP block size gets returned. The **REMOTE** "information only" **BLOCKsize** qualifier contains this new value. The accepted range of values for **BLOCKsize** on a z/OS system is 2048 to 65535 bytes. Most systems will only support a maximum **BLOCKsize** of 32768. The z/OS default is 16384. The minimum spelling is -BLOCK.

**-COMmand** (optional) startup command file name that may be used by the host in which you are attempting to connect. The minimum spelling is -COM.

**-INTerval** (optional) connect retry interval in seconds used for connection retries when connecting to a remote host. The accepted range of values for **INTerval** is 1 to 60 seconds. The default is to retry every 5 seconds. The minimum spelling is -INT.

**-PASSword** (optional) default login password that is used to validate a user on a remote host during a connect. **PASSword** is usually overridden on the command line of the **CONnect**

command. This qualifier must be used in conjunction with the **-USERname** qualifier. The minimum spelling is -PASS.

**-PROFile** (optional) startup profile file name that may be used by the remote host during a connect. The minimum spelling is -PROF.

**-PROJect** (optional) login project code that may be used by the remote host during a connect. The minimum spelling is -PROJ.

**-QUIet** (optional) tells NetEx/eFT whether or not to display intermediate connect messages. This value should be set to either ON or OFF. The default is OFF. The minimum spelling is -QUI.

**-SCRIpt** (optional) script file name that is used by some remote hosts during the connect and login process. The minimum spelling is -SCRI.

**-SEArch** (optional) describes the server startup files to be read during connect time. Refer to "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106 for more information. The default is "(SITE) (USER)". The minimum spelling is -SEA.

**-SECondary** (optional) secondary login password that can be used by the remote host during a connect. The minimum spelling is -SEC.

**-SECUre** (optional) Secure connection. It can only be used if the Secure feature is licensed, the connection is IP (e.g. not NetEx) and the Secure feature is supported/licensed by the remote host. The minimum spelling is -SECU.

**-SERvice** (optional) service name that NetEx/eFT tries to connect to on the remote host during a connect. This service name is EFT by default. The minimum spelling is -SER.

**-SITE** (optional) site-specific login information that may be used by the remote host at connect time.

**-SSLCIPHERS** specifies which ciphers to use. It is a list of four-digit numbers. If omitted, all ciphers are available. See IBM's "z/OS Cryptographic Services System SSL Programming" for the four-digit codes

**-SSLPROTOCOL** specifies which protocols to use. "ALL" (default) allows all protocols to be used. "ALL:-TLSv1" says all protocols except tlsv1 may be use. The valid protocols are: ALL, TLSV1, TLSv1.1 TLSv1.2. Addition may also be use like: TLSv1.1+TLSv1.2

**-TIMeout** (optional) connect timeout value in seconds. If a connection cannot be established within **TIMeout** seconds you will receive an error message from NetEx/eFT. The default value is 2 minutes. The accepted range of values is 0-32767. The minimum spelling is -TIM.

**-USERname** (optional) default login name of the user attempting to connect. **USERname** is usually overridden on the command line of the **CONnect** command. This qualifier must be used in conjunction with the **-PASSword** qualifier. The minimum spelling is -USER.

**-VERBose** (optional) when this qualifier is set to ON, login information returned from the remote host is displayed to the local user. When this qualifier is OFF, the login information is not displayed. The default is ON. The minimum spelling is -VERB.

**host** (required) name of the remote host to which you want to connect.

**username** (optional) your login username on the remote host. This is the positional version of the **USERname** qualifier.

**password** (optional) your login password on the remote host. This is the positional version of the **PASSword** qualifier.

**arg** (optional) any number of argument strings that get passed along to the remote host at connect time.

## Host Dependencies

Many of the **CONnect** qualifiers are treated differently depending on the host to which you are connecting. Also, the optional arguments are both host- and site-dependent. Refer to the remote host's "Remote User's Guide" for further detail.

## Examples

To securely connect to a host named "zpdt2" with the username "test1" and the password "test1":

```
eFT> conn zpdt2 test1 test1 -serv efts -secure
eFT: Connected to service 'efts' on host 'zpdt2'.
 ================================================================================
 ICH70001I TEST1    LAST ACCESS AT 12:56:54 ON WEDNESDAY, NOVEMBER 13,
2019
 IKJ56455I TEST1 LOGON IN PROGRESS AT 13:51:42 ON NOVEMBER 13, 2019
 IKJ56951I NO BROADCAST MESSAGES
 READY
 ================================================================================
 eFT: Logged in as user 'test1'.
 eFT:
 eFT: Welcome to MVS NTXeFT version 5.5.0-9701 N24
 eFT:
zpdt2> show host
eFT:
eFT: active --> (1) Host=zpdt2    Secure=ON    User=TEST1
eFT:
zpdt2>
```

To connect to a host named "sun" in the NCT[2] with the username "smith" and the password "allen", type:

```
eFT> connect sun smith allen
eFT: Connected to Service Initiator on host 'SUN'
eFT: Logged in as user 'smith'.
eFT: Connected to service 'EFT31' on host 'SUN'
```

To connect to host "vax" with the username "jones", the password "jane", block size of 4096 bytes, and connect timeout of 10 seconds, you would type:

```
eFT> connect -blocksize 4096 -timeout 10 -
More>> vax jones jane
eFT: Connected to Service Initiator on host 'VAX'
=====================================================
             Welcome to VAX VMS
                *     *     *
=====================================================
eFT: Logged in as user 'jones'.
eFT: Connected to service 'EFT01' on host 'VAX'
```

[2] NCT – Network Control Table is a feature of the NetEx/IP software products.

An alternative way of setting the **BLOCKsize** and **TIMeout** qualifier values would be to issue the following:

```
eFT> set connect blocksize 4096
eFT> set connect timeout 10
eFT> connect vax jones jane
eFT: Connected to Service Initiator on host 'VAX'
==================================================
             Welcome to VAX VMS
               *      *     *
==================================================
eFT: Logged in as user 'jones'.
eFT: Connected to service 'EFT01' on host 'VAX'
```

This second approach would cause the **CONnect** default values to be changed for all subsequent connects during this NetEx/eFT session whereas the first approach would only affect the connect being issued.

## Related Topics

DISconnect Command
SET HOst Command
SHow HOst Command

# CONTinue Command

## Description

The **CONTinue** command is a no-op command. Its most useful purpose is to provide an action for the **ON** command when a particular exception is to be ignored.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| CONTinue | | |

Where:

**CONTinue** (required) is the verb for this command. The minimum spelling is CONT.

## Example

In this example the **CONTinue** command is used as the action part of an **ON ERRor** command. In the following script any errors are completely ignored:

```
* Sample EFT script - this script
* continues should any EFT error occur
*
on error continue
set variable count 1
LOOP:
send file{count}
set variable count {inc(count)}
{le(count, 5, "goto LOOP")}
```

## Related Topics

ON Command

# DISconnect Command

## Description

Terminate the connection from the current remote host which was previously connected with **CONnect**. Following a **DISconnect**, you will not have any "active" remote connections even though you may still be connected to other hosts. Use the **SHOW HOST** command to display all current remote connections. The **SET HOST** command can be used to make an idle connection active again.

An implied disconnect of all connections takes place following an **EXIT** or **QUIT** command from a NetEx/eFT session.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| DISconnect | [-QUIet \| ON \| ]<br>\| OFF \| | |

Where:

**DISconnect** (required) the verb for this command. The minimum spelling is DIS.

**-QUIet** (optional) controls whether disconnection from a host is indicated with a message or not. This value should be set to either ON or OFF. The default is OFF. The minimum spelling is -QUI.

## Examples

Assume connections have already been made to the hosts "VX1" and "IBM". This results in the following output from **SHOW HOST**:

```
eFT> show host
eFT:              (1) Host=VX1      User=scott
eFT: active --> (2) Host=IBM      User=meyers
```

A **DISconnect** at this point terminates the current "active" connection (host IBM) as seen below:

```
eFT> disconnect
eFT: Disconnected from host IBM.
eFT> show host
eFT:              (1) Host=VX1      User=scott
```

The connection to IBM has been terminated and only the connection to host vx1 remains (still idle). To disconnect from it, you would have to use **SET HOST**, then **DISconnect** as seen below:

```
eFT> set host vx1
eFT> disconnect -quiet
```

When the **QUIet** qualifier is used, the message confirming disconnect is not displayed.

## Related Topics

CONnect Command
SET HOst Command
SHow HOst Command

# EXit Command

## Description

The **EXit** command causes NetEx/eFT to exit to the previous input level. When **EXit** is issued within an input script, execution of the input script is stopped and control is returned to the previous input level (either an input script or interactive command line). **EXit** differs from **QUIT** in that it returns control only to the previous input level. **QUIT** always returns control to the interactive input level (command line).

When issued from the interactive input level, **EXit** causes NetEx/eFT to terminate. If issued from an input script as part of a noninteractive NetEx/eFT session (e.g., a batch job), the session terminates.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| EXit | | [status] |

Where:

**EXit** (required) the verb for this command. The minimum spelling is EX.

**status** (optional) the value to be returned by an input script, or NetEx/eFT if used at the interactive level. The valid values for status are: Success, Warning, Error, and Fatal. The **ON ERROR** command can be used to capture an error resulting from an input script exiting with a status of Warning or Error. An exit status of Fatal causes NetEx/eFT to immediately abort. If status is not specified, an exit status of Success is assumed.

**Note:** A fatal exit from nested scripts while running under TSO may cause some files to be left open preventing them from being free'd until logoff.

## Examples

To exit the NetEx/eFT session and return to your local system's command line interpreter, you would type:

```
eFT> exit
READY
```

At this point you should receive the prompt you normally receive from your system's command line interpreter (e.g., READY).

The following script exits with an Error status if any error occurs within the script, otherwise the script exits with a status of Success:

```
* Sample NetEx/eFT script
*
on error exit error
send -crc sample.file
exit success
```

If the SEND command above results in an error, the script exits with a status of Error, otherwise it exits with a status of Success.

## Related Topics

DISconnect Command
Quit Command

# GOTO Command

## Description

The **GOTO** command instructs NetEx/eFT to continue processing at the given label. The label must be the first item to appear on the NetEx/eFT command line and must be followed by a colon (":"). All labels are case sensitive and must appear somewhere within the current input level.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| GOTO | | label |

Where:

**GOTO** (required) is the verb for this command.

**label** (required) an alphanumeric string from one to twenty characters in length.

## Examples

The following is an example of a simple loop alias:

```
* Sample GOTO/Label script.  Send 5 files
* having the names FILE1 thru FILE5
*
set variable count 1
LOOP:
send file{count}
set variable count {inc(count)}
{le(count, 5, "goto LOOP")}
*
text All files sent.
```

## Related Topics

INput Command

ON Command

# HELp Command

## Description

The NetEx/eFT **HELp** facility gives you online access to NetEx/eFT topics including host-specific qualifier information (both locally and remotely), the formats and descriptions of all NetEx/eFT commands, and examples of how to use them. Some help text is retrieved from the remote host and therefore requires a remote connection. Typing **HELp** without a topic name will generate a top-level help message followed by a list of all topics and commands for which help is available.

You can abbreviate any topic name on the **HELp** command line (including NetEx/eFT commands) although the abbreviation must be unique to the topic name itself. The unique portion of the topic is represented in uppercase letters as shown in the subtopics list.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| HELp | [-SEArch string] | [topic [subtopic]] |

Where:

**HELp** (required) the verb for this command. The minimum spelling is HEL.

**-SEArch** (optional) allows the user to search alternate paths for NetEx/eFT help files. For more information, refer to "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106. The default is (SITE). The minimum spelling is -SEA.

**topic** (optional) name of a NetEx/eFT topic or command in which you desire additional information.

**subtopic** (optional) a subtopic for which further help is available. These subtopics are displayed in the top-level help information.

## Examples

To get the highest level of help, you would just type:

```
eFT> help
```

This will provide you with a list of all topics in which help is available. From that list you can get help on more specific items of interest. For example, if you want help on the **SEND** command, you would type:

```
eFT> help send
```

Now, depending on the subtopics of **SEND** available, you might type:

```
eFT> help send example
```

which would display a sample **SEND** command.

The **SEArch** qualifier for the **HELP** command is used to define where the NetEx/eFT help files exist:

```
eFT> show help search
eFT:
eFT: SEArch .......... (SITE)
eFT:
```

The default is "(SITE)". When "(SITE)" is not followed by a file in the partitioned data set member,

"efthelp" is assumed. Suppose a user help file is defined, such as "myhelp", which contains the help text for the **LOGIN** alias. The alias **LOGIN** is defined as:

```
eFT> show alias login
eFT:
eFT: LOgin .......... {ndf(1, "ask -prompt ""Hostname? "" 1")}
eFT:                  {ndf(2, "ask -prompt ""Username? "" 2")}
eFT:                  {ndf(3, "ask -secure -prompt "Password? "" 3")}
eFT:                  {ndf(4, "ask -prompt "Qualifiers? "" 4 5 6 7 8 9")}
eFT:                  connect {param(1,9)}
eFT:
```

The **HELp** command does not find the text for the **LOGIN** alias unless the user help file, "myhelp", has been included on the **HELp SEARCH** path. In this case the help file "myhelp" is a partitioned data set member in the "(SITE)" location:

```
eFT> help login
eFT: Help is not available for 'login' (UA-4301)
```

Now add to the HELP SEARCH path "(SITE)myhelp":

```
eFT> set help search {search:help} (SITE)myhelp
eFT> show help search
eFT:
eFT: SEArch .......... (SITE) (SITE)myhelp
eFT:
eFT> help login
eFT:
eFT: FORMAT
eFT:
eFT:     LOGin
eFT:
eFT: DESCRIPTION
eFT:
eFT:     The LOGin alias is used to prompt user's for the
eFT:     necessary LOGin information.
eFT:
```

Now when the **HELp** command is used, two locations, "(SITE)" and "(SITE)myhelp", are searched for help on the requested topic, where "myhelp" is a partitioned data set member in the "(SITE)" location.

# INput Command

## Description

The **INput** command instructs NetEx/eFT to take its commands from the specified input file on the local host. This file may contain any number of NetEx/eFT commands. These commands can be structured in such a way that a sophisticated user could create predefined NetEx/eFT procedures that can be used by beginning NetEx/eFT users. These procedures can prompt users for input, give them instructions, and issue NetEx/eFT commands for them.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| INput | [-CONTinue \| ON \|]<br>\| OFF \|<br>[-ECHO \| ON \|]<br>\| OFF \|<br>[-IGNore \| ON \|]<br>\| OFF \|<br>[-PROMpt string]<br>[-PROMpt2 string]<br>[-SEArch string]<br>[-VERify \| ON \|]<br>\| OFF \| | [source] [arguments] |

Where:

**INput** (required) the verb for this command. The minimum spelling is IN.

**-CONTinue** (optional) tells NetEx/eFT how to respond to an error encountered when processing in-put files. This value should be set to either ON or OFF. ON tells NetEx/eFT to continue processing even if an error is encountered while processing commands in the input file. OFF says to terminate processing of the input file if an error is encountered. The initial setting is OFF. The minimum spelling is -CONT.

**-ECHO** (optional) tells NetEx/eFT whether or not to echo input to the terminal as it reads input commands. Commands are echoed as they appear in the input file, before string substitution (and alias translation) is performed. This value should be set to either ON or OFF. The initial setting is OFF.

**-IGNore** (optional) indicates whether or not to cause input failure if the file name specified is missing or invalid. The default is ON. The minimum spelling is –IGN.

**-PROMpt** (optional) the string used as the NetEx/eFT command prompt. The initial setting is “eFT> ”. The minimum spelling is -PROM.

**-PROMPT2** (optional) a secondary NetEx/eFT command prompt string used for command continuation. The initial setting is “More>> ”.

**-SEArch** (optional) search path used for default INPUT commands (the location of input files). **SEArch** is only used if NetEx/eFT cannot locate the source file specified on the command line. A **SEArch** path is a space-separated list of z/OS file specifications. If **SEArch** is defined, NetEx/eFT will use it to locate input files. Refer to “IBM z/OS NetEx/eFT SEARCH

Keywords (SITE), (USER), and (NONE)" on page 106 for more information. The minimum spelling is -SEA.

**-VERify** (optional) works like **ECHO** but displays input commands after string substitution (and alias translation) has taken place. This value should be set to either ON or OFF. The initial setting is OFF. The minimum spelling is -VER.

**source** (optional) the file specification for the input file on the local host. NetEx/eFT attempts to open this file before using the **INput SEArch** path.

**arguments** (optional) zero or more arguments that are passed to the input file as positional parameters for parameter substitution.

Using the **INput** command causes a new **INput** environment to be established. The values for all the above qualifiers are initialized to the then current values. Changing a qualifier value changes the value for the duration of the input script only. Exiting the input script restores the **INput** qualifier values to the values existing before the **INput** command was issued.

## Examples

To input NetEx/eFT commands from a sequential data set named "setup.si" located in the current local TSOPREFIX, you would type:

```
eFT> input -echo on setup.si
```

NetEx/eFT will then attempt to execute all of the commands in 'setup.si' before displaying the interactive "eFT> " prompt again. The "-echo on" switch forces each line from the input file to be echoed to the screen as it is processed.

If "setup.si" was set up to accept positional parameters, you could also pass arguments on the **INput** command line as:

```
eFT> input setup.si HOST3 Smith
```

Suppose there exists a file by the name of "myalias.ua" in the user's login directory. By default, there is no **INput SEArch** path defined. If a search path is defined as follows:

```
eFT> set in search (USER)*.ua
```

The user need only specify the filename portion of the file specification if the files extension is ".ua", and the file is located in the user's default TSOPREFIX:

```
eFT> input myalias
```

The "input myalias" command uses the **SEArch** path to find the NetEx/eFT script file "myalias.ua" in the user's default TSOPREFIX. Notice that the **INput SEArch** qualifier is defined as "(USER)*.ua", the '*' is replaced by the source name to the input command, in this case "myalias". The order of processing for an **INput** file is: check the TSOPREFIX for a file with the name "prefix.MYALIAS", and then check the search path ("prefix.MYALIAS.UA").

The user can also execute an input file without preceding the name of the input file by the **INput** command. The order of processing a command is: check for alias, check for command, and check for an input file on the search path. Setting the search path as in the previous example:

```
eFT> set input search (USER)*.ua
```

The "myalias.ua" script file can be executed by issuing the command:

```
eFT> myalias
```

Command processing checks for an alias by the name “myalias”, then a NetEx/eFT command by the name “myalias”, and then uses the **INput SEArch** qualifier to look for the sequential data set “myalias.ua” in the user’s default TSOPREFIX.

## Related Topics

OUTput Command
SET ALias Command

# LOCal Command

## Description

**LOCal** executes a command on the local z/OS host and displays the results. The command can be a valid TSO/E command or CLIST, or an alias command defined using **SET LOCal ALIAS**, or one of the predefined host independent commands (e.g., **DIRECTORY**, **TYPE**, etc.). Whatever the case may be, the command specified must translate into a valid TSO/E command or CLIST on the z/OS host or it will return an error to you.

NetEx/eFT for IBM z/OS supports the ISPF "ISPEXEC" command interface in ISPF 2.2 or higher for **LOCal** command execution. NetEx/eFT must be invoked as an ISPF application. The z/OS NetEx/eFT client CLIST will invoke the client as an ISPF application if the user is running under a valid ISPF environment.

If the command parameter is missing, you will enter an interactive local terminal mode. You will remain in local terminal mode until you leave it by typing "EXIT", at which time you will again see the "`eFT>` "prompt. All remote host connections will still be intact.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `LOCal` | | `[command]` |

Where:

**LOCal** (required) the verb for this command. The minimum spelling is LOC.

**command** (optional) a valid local host command or local alias command.

## Informational Qualifiers

The following informational qualifiers are provided to give the user information about the local z/OS environment. With the exception of **DIRectory**, these qualifiers cannot be modified by the user.

**COPYRight** (string) copyright information for the current release of software.

**DIRectory** (string) the current working directory on the local IBM/z/OS host. This qualifier can be modified by using the **SET LOCal DIRectory** command. This qualifier is the same as **TSOPREfix**.

**GATEway** (string) not used for z/OS.

**HOSTCODE** (string) the native host character code (EBCDIC).

**HOSTENV** (string) the execution environment of the EFT Client or Server. For z/OS it is one of the following values:

**BATCH JOBSTEP**
**STARTED TASK**
**TSO FOREGROUND**
**TSO BACKGROUND**
**UNKNOWN**

| | |
|---|---|
| **HOSTOS** | (string) the level of the host operating system. For z/OS this also contains the MVS FMID. |
| **HOSTTYPE** | (string) Operating system type (z/OS). |
| **NETwork** | (string) network type. |
| **PID** | (integer) z/OS Process ID. |
| **PRODuct** | (string) NESi Product number (EFT213). |
| **STATus** | (string) exit status of the last **LOCal** command. |
| **TSOPREfix** | (string) the current working directory on the local IBM/z/OS host. This qualifier can be modified by using the **SET LOCal TSOPREfix** command. This qualifier is the same as **DIRectory**. |
| **USERname** | (string) the current TSO/E username. |
| **VERSION** | (string) NetEx/eFT version number. |

## Examples

To display all data sets on your local z/OS host with a TSO/E prefix matching the **LOCal** qualifier **TSOPREfix** (assume your current **TSOPREFIX** is "EFTFILE"), issue the TSO/E list catalog ("LISTCAT") command:

```
eFT> local listcat
IN CATALOG:CAT.MV53.TSO
EFTFILE.ISPF.ISPROF
EFTFILE.LOCAL.LIST
EFTFILE.PROGRAM.LIST
EFTFILE.SPFLOG1.LIST
EFTFILE.TEST.FORTRAN
EFTFILE.TEST.LOAD
```

Since there is a host independent command (defined as a local alias) called **DIRectory** that accomplishes the same task as "LISTCAT", you could have also typed:

```
eFT> local dir
```

to get the same results.

To get a listing of all members in an existing partitioned data set named "TEST.FORTRAN", invoke the NetEx/eFT alias **MEMber** as:

```
eFT> local member test.fortran
EFTFILE.TEST.FORTRAN

--RECFM-LRECL-BLKSIZE-DSORG-CREATED--EXPIRES--SECURITY
    FB    80    9040    PO   04/02/88    * *    NONE
--VOLUMES--
  EFTLIB
--MEMBERS--
  PROGRAMA
  PROGRAMB
  SUBROUTS
  TSOFILE
  XTRASUB
```

The same result could have been accomplished with the command:

```
eFT> local LISTDS TEST.FORTRAN MEMBERS HISTORY
```

To delete a data set named “OLDFILES.TEST.LIST”, you would type:

```
eFT> local delete 'OLDFILES.TEST.LIST'
IDC0550I ENTRY (A) OLDFILES.TEST.LIST DELETED
```

To drop into interactive TSO/E on the local IBM z/OS system without losing remote host connections, you can simply type:

```
eFT> local
Type 'EXIT' to return to USER prompt

EFT READY
```

First you are reminded to type “EXIT” when you want to go back into NetEx/eFT, then you are prompted by TSO/E with the prompt EFT READY. At this point you can interact with TSO/E in the usual manner until you again want to return to your NetEx/eFT session. To do this, just exit TSO/E by typing “EXIT”.

```
EXIT
eFT>
```

## LOCAL Command Status

The completion status of the **LOCAL** TSO/E command is placed in the NetEx/eFT variable **{status:local}**. The possible values are:

| | |
|---|---|
| **0** | Normal (successful) completion. |
| **N** | Integer value for unsuccessful completion. Set by TSO/E command. |
| **Sxxx** | Abnormal completion with z/OS system ABEND xxx. |
| **Uxxxx** | Abnormal completion with user ABEND xxxx. |
| **Interrupt** | Unsuccessful completion due to a keyboard interrupt (attention). |
| **Error** | Unsuccessful completion due to invalid, unknown, or unsupported TSO/E command. |

## Related Topics

REMote Command
SET Command

# ON Command

## Description

The **ON** command allows users to catch any one of the exceptions:

| | |
|---|---|
| **ERRor** | on NetEx/eFT error |
| **INTerrupt** | on keyboard interrupt |
| **LOCal_error** | on **LOCAL** command error |
| **REMote_error** | on **REMOTE** command error |

The **ON** command initializes the exception by specifying an action that should occur each time the exception takes place. To turn off the exception handler, issue the same command without an action.

### ON ERRor

The **ON ERRor** exception establishes an alternative action to be taken when a NetEx/eFT error occurs. Without an **ON ERRor** specified, NetEx/eFT terminates all input levels (for nested input scripts) and begins processing at the interactive level. If **INPut CONTinue** is "ON", NetEx/eFT displays the error and continues processing the next command.

### ON INTerrupt

The **ON INTerrupt** exception establishes an alternative action to be taken when a keyboard interrupt occurs. Without an **ON INTerrupt** specified, NetEx/eFT terminates all input levels (for nested input scripts) and begins processing at the interactive level. If **INPut CONTinue** is "ON", NetEx/eFT terminates the current level and continues processing in the next level up.

### ON LOCal_error

The **ON LOCal_error** exception establishes an alternative action to be taken when a **LOCal** command error occurs. A local error occurs when a z/OS command returns any nonzero status.

### ON REMote_error

The **ON REMote_error** exception establishes an alternative action to be taken when a **REMote** command error occurs. A remote error occurs when the remote command issued returns an unsuccessful status. The definition of **REMote_error** is dependent upon the remote host. Some hosts cannot detect command execution errors, in which case **ON REMote_error** becomes ineffective.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `ON` | | `exception [action]` |

Where:

**ON** is the keyword for this command.

**exception** is any one of the following: **ERRor**, **INTerrupt**, **LOCal_error**, or **REMote_error**.

**action** is any NetEx/eFT command or alias.

## Examples

The following is a short NetEx/eFT script that immediately exits should any error occur:

```
* Sample EFT script - this script
* exits should any EFT error occur
*
on error exit
set variable count 1
LOOP:
send file{count}
set variable count {inc(count)}
{le(count, 5, "goto LOOP")}
*
text All files sent.
```

The exception being handled is **ERRor** and the action to be taken, should an error occur, is the NetEx/eFT command **EXit**.

The following is a short NetEx/eFT script that immediately continues execution should any keyboard interrupt occur:

```
* Sample EFT script - this script
* continues should any keyboard interrupt
* occur.
*
on interrupt continue
set variable count 1
LOOP:
send file{count}
set variable count {inc(count)}
{le(count, 5, "goto LOOP")}
*
text All files sent.
```

The exception being handled is **INTerrupt** and the action to be taken should a keyboard interrupt occur is the NetEx/eFT no-op command **CONTinue**.

The following script issues a “loc directory” command with a file name as an argument. If the command fails, implying the file does not exist, the **ON LOCal_error** action is to receive that file:

```
* Sample EFT script
*
* Setup ON LOCal_error, the action is to
* receive 'file{count}'.
*
on LOCal_error {}receive file{count}
set variable count 1
LOOP:
loc -quiet directory file{count}
set variable count {inc(count)}
{le(count, 5, "goto LOOP")}
*
exit
```

## Related Topics

GOTO Command
INput Command

# OUTput Command

## Description

The **OUTput** command instructs NetEx/eFT to capture standard output to a file on the local host. **OUTput** gives the user the capability of saving the output of a local or remote command execution to a file on the local host.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| OUTput | [-COLumns integer]<br>[-CREate \| APPend \|]<br>\| DELETE \|<br>\| NEW \|<br>\| REPlace \|<br>[-FORmat string]<br>[-HOLD \| ON \|]<br>\| OFF \|<br>[-LINes integer]<br>[-PREFix string]<br>[-QUIet \| OFF \|]<br>\| ON \|<br>[-TRUNcate \| OFF \|]<br>\| ON \|<br>[-UNIT string]<br>[-VOLUME string] | [destination] |

Where:

**OUTput** (required) the verb for this command. The minimum spelling is OUT.

**-COLumns** (optional) the maximum number of columns per terminal page of output. This represents the maximum number of characters across a page or terminal screen. For z/OS, the default is your terminal line size minus 2. The minimum spelling is -COL.

**-CREate** (optional) describes how to create the output file on the local system. The valid values are **APPend**, **DELete**, **NEW**, and **REPLace**. The default is **REPLace**. Refer to "IBM z/OS File Transfer Qualifiers and Default Values" on page 41 for more information on **CREate** qualifier values. The minimum spelling is -CRE. When "TAPE" is specified and a **-VOLSER** is specified then "-CREate REPLace" must be specified. Otherwise, when a **-VOLSER** is not specified then "-CREate NEW", "-CREate REPLace", or "-CREate DELete" may be specified.

**-FORmat** (optional) all NetEx/eFT messages are displayed using the format string defined by this qualifier. The **msg()** string function is used to construct an appropriate format for NetEx/eFT messages. The minimum spelling is -FOR.

**-HOLD** (optional) suspends scrolling of the output from a command or input file. The number of lines that scroll by before the output is suspended is specified by the **LINes** qualifier.

**-LINes** (optional) the maximum number of lines per terminal page of output. For z/OS, the default is the number of lines on your terminal. The minimum spelling is -LIN.

**-PREFix** (optional) the prefix string displayed before each line of NetEx/eFT output. The default is "`eFT: `". The minimum spelling is -PREF.

**-QUIet** (optional) when this qualifier is "ON", no NetEx/eFT output is displayed to the user's terminal. If an output destination file exists, the output is still captured to the destination file. The default is OFF. The minimum spelling is -QUI.

**-SPAce** (integer) the number of bytes which NetEx/eFT should allocate to the output file. For z/OS, the default is 16384 (16 Kbytes). The minimum spelling is -SPA.

**-TRUNcate** (optional) works in conjunction with the **COLumns** qualifier. If any NetEx/eFT output lines are longer than the **COLumns** value, the lines are truncated when this qualifier is "ON". The default is OFF. The minimum spelling is -TRUN.

**-UNIT** (optional) tells EFT to create the data set on a specific device or certain type or group of devices. The UNIT qualifier must be a 1- to 8-character alphanumeric string including national characters ($, #, or @) where the first character cannot be a digit. If no unit name is specified, the unit name in the TSO/E user's profile is used. When "TAPE" is specified and a **-VOLSER** is specified then "-CREate REPLace" must be specified. Otherwise, when a -**VOLSER** is not specified then "-CREate NEW", "-CREate REPLace", or "-CREate DELete" may be specified.

**-VOLUME** (optional) tells EFT to create the data set on a specific volume. The VOLUME qualifier must be a 1- to 8-character alphanumeric string including national characters ($, #, or @) where the first character cannot be a digit. There is no default VOLUME name. When "TAPE" is specified and a **-VOLSER** is specified then "-CREate REPLace" must be specified. Otherwise, when a -**VOLSER** is not specified then "-CREate NEW", "-CREate REPLace", or "-CREate DELete" may be specified.

**destination** (optional) a local file specification that will receive the captured NetEx/eFT output. Omitting the destination file specification causes the current output file (if any) to be closed.

## Informational Qualifiers

The following qualifiers are provided to give the user information about the OUTput command.

**DESTination** (string) the output destination file specification.

## Examples

To begin capturing NetEx/eFT output to a file on the local host named session, you would type:

```
eFT> output session
```

Now every line of NetEx/eFT output that appears on the screen will also be sent to the output file until you close the file with another output command:

```
eFT> output
```

To tell NetEx/eFT to hold the screen every time a full screen of output is displayed, type the following:

```
eFT> set output hold on
```

## Related Topics

INput Command

# Quit Command

## Description

The **Quit** command causes NetEx/eFT to return control to the interactive (command line) input level. When **Quit** is issued from a nested input script, control is returned all the way back to the interactive input level (i.e., any input scripts nested before the one issuing the **Quit** are also terminated). **Quit** differs from **EXit** in that **Quit** always returns control to the interactive level whereas **EXit** returns control back to the previous input level, whether it was interactive or another input script.

When issued from the interactive input level, **Quit** causes NetEx/eFT to terminate. If issued from an input script as part of a noninteractive NetEx/eFT session (e.g., a batch job), the session terminates.

The **Quit** command can set the following MVS condition codes:

**Quit** Issuing **Quit** with no operands sets the condition code based upon the completion status of the prior NetEx/eFT command.
**Quit Success** Sets condition code zero (0).
**Quit Warning** Sets condition code four (4).
**Quit Error** Sets condition code eight (8).
**Quit Fatal** Sets condition code twelve (12).

No other condition code values are supported at this time.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `Quit` | | `[status]` |

Where:

**Quit** (required) the verb for this command. The minimum spelling is Q.

**status** (optional) the value to be returned by an input script, or NetEx/eFT if used at the interactive level. The valid values for status are: Success, Warning, Error, and Fatal. The **ON ERROR** command can be used to capture an error resulting from an input script exiting with a status of Warning or Error. An exit status of Fatal causes NetEx/eFT to immediately abort. If status is not specified, an exit status of Success is assumed.

## Examples

If you desire to leave NetEx/eFT and return to your local system's command line interpreter, you would type:

```
eFT> quit
READY
```

At this point you should receive the prompt you normally receive from your system's command line interpreter (e.g., READY).

If you desire to leave NetEx/eFT and return an Error status to your local system, you would type:

```
eFT> quit error
READY
```

At this point you should receive the prompt you normally receive from your system's command line interpreter (e.g., READY).

## Related Topics

DISconnect Command
EXit Command

# RECeive Command

## Description

The **RECeive** command receives the source file or data set from the current active remote host and saves it as the destination file or data set on the local host.

If the destination data set specification is not a fully qualified data set name (i.e., if it is not enclosed in single quotes) or DD name specification, the data set name is prefixed with the current default value of **DIRectory** (which is equivalent to TSOPREFIX) on the destination z/OS host. Similarly, if the source file specification is not fully qualified (if connected to another z/OS host), or if no path to the file is specified (i.e., if a file name is given without a directory on non-z/OS hosts), the file is prefixed with the default value of **DIRectory** on the source z/OS host. In either case, **DIRectory** is used as the path to the file or data set name.

If the destination parameter is not specified at all, a data set name will be constructed from the source file name and will be prefixed with the value of **DIRectory** (TSOPREFIX) on the destination z/OS host. The source file name and the extension will both be truncated to eight characters if needed.

NetEx/eFT for IBM z/OS will transfer into either a sequential or a partitioned data set. If a member name is supplied as part of the data set name, NetEx/eFT will assume the data set is partitioned, and either create a new partitioned data set and member, or create/replace the member in an existing partitioned data set. If a member name is supplied and the existing destination data set is sequential, an error message is returned. If no member name is supplied and the existing destination data set is partitioned, NetEx/eFT will build a member name from the source file (data set) name and create or replace the member. Otherwise, NetEx/eFT will create a new sequential data set or replace (overwrite) an existing sequential data set.

Experienced z/OS users may also use a DD Name (Data Definition Name) in place of a z/OS data set name so long as the DD Name references an allocated data set with valid characteristics for the file transfer. The syntax for specifying a DD name is DD:ddname or DD:ddname(member). z/OS secondary space requests will not be processed if the member name is provided on a DD specification.

The source file name may include the NetEx/eFT wildcard characters ***** and **?**. See "Source Wildcard Support for IBM z/OS File Transfers" on page 56 and "Destination Wildcard Support for IBM z/OS File Transfers" on page 58 for more details.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `RECeive` | `[-qualifiers]` | `source [destination]` |

Where:

**RECeive** (required) the verb for this command. The minimum spelling is REC.

**-qualifiers** (optional) the qualifiers that apply to the **RECeive** command. Refer to "File Handling Under IBM z/OS NetEx/eFT" on page 41 for a description of the **RECeive** command qualifiers. See also the file handling section of the remote host manual for details on qualifiers supported.

**source** (required) the file specification for the file on the remote host that you intend to receive.

**destination** (optional) the file specification for the new file that is to be created on the local host.

## Examples

Refer to “File Handling Under IBM z/OS NetEx/eFT” on page 41 for examples of using the **RECeive** command.

## Related Topics

LOCal Command
REMote Command
SENd Command
SET Command

# REMote Command

**Note:** For details on how the **REMote** command operates, refer to the User's Guide for the remote host. The description provided here gives a general overview of the command from the perspective of the local NetEx/eFT initiator.

## Description

**REMote** executes a command on the remote host and displays the results on the local system. The command can be a valid command for the remote host's command line interpreter, an alias command defined using **SET REMOTE ALIAS**, or one of the predefined host independent commands (e.g., **DIRECTORY**, **WHO**, **TYPE**, etc.). Whatever the case may be, the command specified must translate into a valid command on the remote host or it will return an error.

There is no interactive mode for the **REMote** command. You must specify a remote host command containing all required parameters or an error will result. Also, since **REMote** doesn't operate in an interactive mode, you will not be able to successfully execute commands or programs on the remote host that require an interactive user (e.g., one that prompts for input, such as graphics programs). You could, however, use **REMote** to submit a job file on the remote host (or execute a script) that would accomplish much the same task given proper input data.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| REMote | [ \|-DIRectory \| string]<br>  \|-TSOPREFix \|<br>[-PREFix string]<br>[-QUIet  \| OFF \| ]<br>  \| ON  \| | command |

Where:

**REMote** (required) the verb for this command. The minimum spelling is REM.

**-DIRectory** or **-TSOPREFix** (optional) the current working directory on the remote host. This qualifier can be modified using the **SET REMote DIRectory** command.

**-PREFix** (optional) a prefix string that appears before each line of remote command output. Its purpose is to "flag" output as coming from the remote host versus NetEx/eFT or local output. You can define this qualifier to be null if no prefix is desired. The minimum spelling is -PREF.

**-QUIet** (optional) ON prevents remote command output from being displayed. The default is OFF. The minimum spelling is -QUI.

**command** (required) a valid remote host command or remote alias command.

## Informational Qualifiers

The following qualifiers are provided to give the user information about the remote environment. Most are informational and cannot be modified by the user. **-DIRectory**, **-PREFix**, and some host-specific qualifiers may be modified.

**-BLOCKsize** (integer) NETEX negotiated block size.

**-HOST** (string) the remote host name.

**-HOSTCODE** (string) the native host character code.

**-HOSTTYPE** (string) Operating system type.

**-PID** (integer) Process ID.

**-PRODuct** (string) NESi product number.

**-SERvice** (string) the TCP/IP or NETEX service connected to.

**-STATus** (integer) exit status of the last REMote command.

**-TRANSlate** (string) the current translation in effect.

**-USERname** (string) the username.

**-VERSION** (string) the NetEx/eFT version number.

## Examples

To display all users currently logged into the remote VAX/VMS host you would execute the VMS show users command from NetEx/eFT as:

```
eFT> remote show users
VMS:          VAX/VMS Interactive Users
VMS:           01-JAN-1987 08:00:00.00
VMS:  Total number of interactive users = 2
VMS:
VMS: Username  Process Name    PID            Terminal
VMS: SMITH     JOE SMITH     00000093         TTA6:
VMS: MEYERS    ED MEYERS     0000009A         TTA7:
```

The prefix VMS. identifies results that are being returned from the remote VAX/VMS host. Since there is a predefined host independent command (which is really a remote alias) called **WHO** that accomplishes the same task as show users, you could have also typed:

```
eFT> remote who
```

Assume the remote host is running UNIX. To display what it thinks is the current date, you would type:

```
eFT> remote date
```

The results would be whatever the current date is on the remote host.

## REMote Command Status

The completion status of the **REMote** TSO/E command is placed in the NetEx/eFT variable **{status:remote}**. The possible values are:

| | |
|---|---|
| **0** | Normal (successful) completion. |
| **n** | Integer value for unsuccessful completion. Set by TSO/E command. |
| **Sxxx** | Abnormal completion with z/OS system ABEND xxx. |
| **Uxxxx** | Abnormal completion with user ABEND xxxx. |
| **Interrupt** | Unsuccessful completion due to a keyboard interrupt (attention). |
| **Error** | Unsuccessful completion due to invalid, unknown, or unsupported TSO/E command. |

## Related Topics

LOCal Command
SET Command

# SENd Command

## Description

The **SENd** command sends the source file from the local host and saves it as a destination file or data set on the active remote host.

If the source data set specification is not a fully qualified data set name (i.e., if it is not enclosed in single quotes), the data set name is prefixed with the current default value of **DIRECTORY** (which is equivalent to **TSOPREFIX**) on the destination z/OS host. Similarly, if the destination file specification is not fully qualified (if connected to another z/OS host), or if no path to the file is specified (i.e., if a file name is given without a directory on non-z/OS hosts), the file is prefixed with the default value of **DIRECTORY** on the destination host.

If the destination parameter is not specified at all, a file name will be constructed from the source data set name and will be prefixed with the value of **DIRECTORY** on the destination host.

NetEx/eFT for IBM z/OS will transfer into either a sequential or a partitioned data set. If a member name is supplied as part of the data set name, NetEx/eFT will assume the data set is partitioned, and either create a new partitioned data set and member, or create/replace the member in an existing partitioned data set. If a member name is supplied and the existing data set is sequential, an error message is returned. If no member name is supplied and the existing destination data set is partitioned, NetEx/eFT will build a member name from the source file (data set) name and create or replace the member. Otherwise, NetEx/eFT will create a new sequential data set or replace (overwrite) an existing sequential data set.

Experienced z/OS users may also use a DD Name (Data Definition Name) in place of an z/OS data set name so long as the DD Name references an allocated data set with valid characteristics for the particular file transfer. The syntax for specifying a DD name is “DD:ddname” or “DD:ddname(member)”. z/OS secondary space requests will not be processed if the member name is provided on a DD specification.

The data set name may include the NetEx/eFT wildcard characters ***** and **?**. See “Source Wildcard Support for IBM z/OS File Transfers” on page 56 and “Destination Wildcard Support for IBM z/OS File Transfers” on page 58 for more details.

## Format

Where:

| Command | Qualifier | Parameter |
|---|---|---|
| `SENd` | `[-qualifiers]` | `source [destination]` |

Where:

**SENd** (required) the verb for this command. The minimum spelling is SEN.

**-qualifiers** (optional) the qualifiers that apply to the **SENd** command. Refer to the file handling section of the remote host for details about these qualifiers. For a remote z/OS host, refer to “File Handling Under IBM z/OS NetEx/eFT” on page 41.

**source** (required) the file specification for the file on the local host that you intend to send to the remote host.

**destination** (optional) the file specification for the new file that is to be created on the remote host.

## Examples

Refer to “File Handling Under IBM z/OS NetEx/eFT” on page 41 for examples of the **SENd** command under IBM z/OS. Refer to the file handling section of the appropriate remote host manual for examples under other operating systems.

## Related Topics

LOCal Command
RECeive Command
REMote Command
SET Command

# SET Command

## Description

This form of the **SET** command allows you to change the default value of a command qualifier. Most qualifiers are initially assigned reasonable defaults by NetEx/eFT so novice users can issue commands without being concerned with qualifiers on the command line. Once a user becomes more familiar with NetEx/eFT and wants to perform more complex tasks, he can set up commands with qualifier defaults of his own choosing. This is done with the **SET** command.

The value assigned to a command qualifier with **SET** becomes the new default for the command. The value of a qualifier is the remainder of the **SET** command line following the qualifier parameter. If a value is not specified on the **SET** command line, the qualifier is defined to be nothing (assigned a null value). The qualifier specified must be valid for the command. Use the **SHow QUAlifiers** command to see which qualifiers and values are valid for a given command.

The **value** parameter is taken literally unless it is enclosed in double quotes ("value"). If the value is enclosed in double quotes, NetEx/eFT expects any embedded quotes (that is, within the value) to be "escaped" using two double quotes together. This special processing allows **value** to include leading and/or trailing spaces.

**Note:** The **SET** command is the only way to change the **DIRECTORY** qualifier for the **LOCAL** and **REMOTE** commands.

The commands **SET VARIABLE** and **SET GLOBAL** are detailed in later sections.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `SET` | | `command qualifier [value]` |

Where:

**SET** (required) the verb for this command.

**command** (required) name of a NetEx/eFT command that supports the use of command qualifiers (e.g., **CONNECT**, **REMOTE**, etc.)

**qualifier** (required with command) name of a valid qualifier for the command specified.

**value** (optional) the new default value you are assigning to this qualifier.

## Examples

To change the default INPUT prompt string to ">> ", type:

```
eFT> set input prompt >>
>> show input prompt
eFT: PROMpt ............. >>
```

To change the current **REMOTE** default directory to “DRA2:[TEMP]” (assuming the remote host is a VAX running VMS), you would type:

```
>> set remote directory DRA2:[TEMP]
>> show remote directory
eFT: DIRectory ..........  DRA2:[TEMP]
```

## Related Topics

SHOW Command
SHow QUALifier Command

# SET ALias Command

## Description

The **SET ALias** command allows you to define your own alias commands. By creating aliases, you can tailor your own command language making things very simple for both beginning and advanced NetEx/eFT users.

You can define three types of aliases: NetEx/eFT, local, and remote. NetEx/eFT alias definitions are made up of other NetEx/eFT commands and are invoked simply by typing the alias command name in response to the NetEx/eFT prompt. Local alias definitions are commands understood by your local host's command line interpreter and are invoked by typing **LOCAL** followed by the alias command name. Remote alias definitions are commands understood by the remote host's command line interpreter and are invoked by typing **REMOTE** followed by the alias command name. A remote connection is required to create a remote alias. The remote alias is only in effect for the remote connection that is active when the remote alias is defined.

The definition of the alias is the remainder of the line following the alias command name. You redefine an existing alias by using **SET ALias** to overwrite the previous definition with the new definition. If you do not supply the definition parameter, the alias becomes undefined. Use the **SHow ALias** command to see which aliases are defined.

You can also create multicommand NetEx/eFT aliases using the NetEx/eFT escape character "!" at the end of the line. Multicommand aliases are discussed further in "Creating Multi-command NetEx/eFT Aliases".

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `SET [ \| LOCal \|] ALias`<br>`  \| REMote \|` | | `alias [definition] [!]` |

Where:

**SET** (required) the verb for this command.

**LOCal** or **REMote** (optional) entering either **LOCal** or **REMote** here tells NetEx/eFT to create a local or remote alias command instead of a NetEx/eFT alias. The minimum spellings are LOC and REM.

**ALias** (required) the subject for this command. The minimum spelling is AL.

**alias** (required) name of the new or existing alias in which you are attempting to define. If a portion of this is capitalized, that portion will be the minimum required spelling for the alias.

**definition** (optional) the string definition of the alias command you are defining.

**!** (optional) indicates the alias definition continues on the next line.

## Host Dependencies

**LOCal** and **REMote** aliases should translate to host dependent commands on the local or remote host respectively.

## Examples

To define a NetEx/eFT alias called "hc" to be the NetEx/eFT command "**HELp COMmands**", you would type:

```
eFT> set alias hc help commands
```

Now, to display the NetEx/eFT command summary, you would just type

```
eFT> hc
```

which is equivalent to typing:

```
eFT> help commands
```

If your local host supports the command "whoami", which displays your local username, you can create a local alias command that displays your local username:

```
eFT> set local alias ? whoami
```

To invoke your new alias, all you need to do it type:

```
eFT> local
```

which is equivalent to typing:

```
eFT> local whoami
```

To create a multicommand NetEx/eFT alias named STAtus that displays the current LOCAL and REMOTE qualifier defaults, you would type:

```
eFT> set alias STAtus show local !
More>> show remote
```

To execute your new alias command you would type:

```
eFT> status
```

which would result in the list of **LOCal** qualifier defaults followed by the list of **REMote** qualifier defaults.

## Related Topics

LOCal Command
SHow ALias Command
REMote Command

# SET GLOBal Command

## Description

The **SET GLOBal** command assigns a value to a global variable name. The scope of a global variable within NetEx/eFT scripts is not limited to the input level on which it was defined (global in scope). Refer to the **SET VARiable** command if the scope of the variable needs to be limited to the current input level. If the **value** parameter does not exist the variable becomes undefined.

The **value** parameter is taken literally unless it is enclosed in double quotes ("value"). If the value is enclosed in double quotes, NetEx/eFT expects any embedded quotes (that is, within the value) to be "escaped" using two double quotes together. This special processing allows **value** to include leading and/or trailing spaces.

Global variables may be referenced by placing the variable name between braces, in one of two ways. Assume a global variable by the name of "num" exists. The global variable "num" can be referenced by either "{num}" if a local variable by the same name does not exist, or by "{num:global}" to explicitly request the global definition.

If a global variable is referenced and does not exist, the variable is replaced by a NULL string. If the global variable "num" is not defined the string "{num:global}" is equivalent to " ". The number of globals that can be defined at one time is limited.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| SET GLOBal | | name [value] |

Where:

**SET** (required) the verb for this command.

**GLOBal** (required) the keyword for this command. The minimum spelling is GLOB.

**name** (required) global variable name. NetEx/eFT variable names must be alphanumeric and no longer than 20 characters.

**value** (optional) variable value. The value assigned to the variable name is the remainder of the line starting with the first nonblank character.

## Example

The following sample script assigns a numeric value to the global "num":

```
eFT> set global num 1
eFT> text The global num = {num:global}
eFT: The global num = 1
eFT> show global num
eFT:
eFT: NUM ........... 1
eFT:
```

## Related Topics

SHow GLOBal Command
SET VARiable Command
SHow VARiable Command

# SET HOst Command

## Description

The **SET HOst** command allows you to select a host as the current "active" host. **SET HOst** is typically used following a **DISconnect** command in order to activate some other remote connection that is currently idle.

You can either specify a host name or host number on the command line. The host number is obtained from the **SET HOst** command.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| SET HOst | | hostname |

Where:

**SET** (required) the verb for this command.

**HOst** (required) the subject for this command. The minimum spelling is HO.

**hostname** (required) host name or host number of a previously established connection.

## Examples

Assume the following connections have already been made:

```
eFT> connect vaxa scott john
eFT> connect sun01 meyers ed
```

Typing the SHOW HOST command then would result in:

```
eFT> show host
eFT:            (1) Host=vaxa      User=scott
eFT: active -->(2) Host=sun01     User=meyers
```

We can use SET HOST here to "reactivate" the first connection (to host vaxa) in one of two ways:

```
eFT> set host 1
```

or

```
eFT> set host vaxa
```

The result would be the same:

```
eFT> show host
eFT: active --> (1) Host=vaxa      User=scott
eFT:            (2) Host=sun01     User=meyers
```

Now any NetEx/eFT command that requires a remote connection (e.g., **SEND** and **RECEIVE**), would communicate with host "vaxa". Note that if two or more sessions exist with one host (e.g., "vaxa"), a **SET HOst (host)** command will activate the lowest numbered session with that host.

## Related Topics

CONnect Command
DISconnect Command
SHow HOst Command

# SET VARiable Command

## Description

The **SET VARiable** command assigns a value to a local variable name. The scope of the variable is limited to the current input level (local in scope). If the variable needs to be visible outside the current input level refer to the **SET GLOBAL** command. If the value parameter does not exist, the variable becomes undefined. A variable may also be set using the **ASK** command.

The **value** parameter is taken literally unless it is enclosed in double quotes ("value"). If the value is enclosed in double quotes, NetEx/eFT expects any embedded quotes (that is, within the value) to be "escaped" using two double quotes together. This special processing allows **value** to include leading and/or trailing spaces.

Variables are referenced by placing the variable name between braces. Assume a variable by the name of "num" exists. The variable "num" can be referenced by "{num}".

If a variable is referenced and does not exist, the variable is replaced by a NULL string. If the variable "num" is not defined, and the global variable "{num:global}" is not defined, the string "{num}" is equivalent to " ".

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `SET VARiable` | | `name [value]` |

Where:

**SET** (required) the verb for this command.

**VARiable** (required) the keyword for this command. The minimum spelling is VAR.

**name** (required) variable name. NetEx/eFT variable names must be alphanumeric and no longer than 20 characters.

**value** (optional) variable value. The value assigned to the variable name is the remainder of the line starting with the first nonblank character.

## Examples

The following sample script assigns a numeric value to the variable "num":

```
eFT> set variable num 1
eFT> text The variable num = {num}
eFT: The variable num = 1
eFT> show variable num
eFT:
eFT: NUM ..... 1
eFT:
```

## Related Topics

ASK Command
SHow VARiable Command
SET GLOBal Command
SHow GLOBal Command

# SHOW Command

## Description

**SHOW** allows you to display the current default for any command qualifier. You can display the current defaults for all qualifiers of a command by leaving the qualifier parameter off the command line.

Note that some commands require a remote connection to show a complete list of qualifiers with the **SHOW** command. These commands, for example **RECEIVE**, gather information from the remote host and may display only a partial list without a connection. They also may just return an error message.

The commands **SHOW GLOBAL** and **SHOW VARIABLE** are described later in this document.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| SHow | | command [qualifier] |

Where:

**SHow** (required) the verb for this command. The minimum spelling is SH.

**command** (required) the name of a NetEx/eFT command that supports the use of command qualifiers (e.g., **CONNECT**, **LOCAL**, etc.).

**qualifier** (optional) name of a valid qualifier for the given command.

## Examples

To display the current default values for all INPUT qualifiers, you would type:

```
eFT> show input
eFT: CONTinue ............ on
eFT: ECHO ................ off
eFT: PROMpt .............. eFT>
eFT: PROMPT2 ............. More>>
eFT: SEARCH ..............
eFT: VERify .............. off
```

If you are only interested in the default value of qualifier PROMPT2, you would type:

```
eFT> show input prompt2
eFT: PROMPT2 .............  More>>
```

## Related Topics

SET Command
ASK Command

# SHow ALias Command

## Description

**SHow ALias** allows you to display the alias command definitions for any previously defined aliases. To display local host aliases, you need to type **SHow LOCal ALias**. To display remote host aliases, you need to type **SHow REMote ALias**. If you do not specify either local or remote, NetEx/eFT aliases will be displayed.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| `SHow [ \| LOCal \|] Alias`<br>`  \| REMote \|` | | `[alias]` |

Where:

**SHow** (required) the verb for this command. The minimum spelling is SH.

**ALias** (required) the subject for this command. The minimum spelling is AL.

**LOCal** or **REMote** (optional) entering either LOCAL or REMOTE tells NetEx/eFT to display a local or remote alias definition rather than a NetEx/eFT alias definition. The minimum spellings are LOC and REM.

**alias** (optional) the name of a previously defined alias command.

## Examples

Suppose the remote alias "WHO" is defined to be a substitute for the remote command "whoami". To display the definition for the remote alias, you would type:

```
eFT> show remote alias who
eFT: WHO ............... whoami
```

To see all NetEx/eFT alias command definitions you would type:

```
eFT> show alias
eFT: VAX ............... connect vax meyers ed
eFT: SL ................ show local
eFT: ? ................. remote who
```

## Related Topics

SET ALias Command

# SHow GLOBal Command

## Description

The **SHow GLOBal** command displays the currently assigned value of the global variable specified. If no specific variable is specified, then all global variables and their values are displayed.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| SHow GLOBal | | [name] |

Where:

**SHow** (required) the verb for this command. The minimum spelling is SH.

**GLOBal** (required) the keyword for this command. The minimum spelling is GLOB.

**name** (optional) global variable name. NetEx/eFT variable names must be alphanumeric and no longer than 20 characters.

## Examples

The following commands define two variables, “first” and “last”:

```
eFT> set global first john
eFT> set global last doe
```

The following **SHow GLOBal** command displays the values of the global variables “first” and “last”:

```
eFT> show global first
eFT:
eFT: FIRST ............. john
eFT:
eFT> show global
eFT:
eFT: FIRST ............. john
eFT: LAST .............. doe
eFT:
```

## Related Topics

SET GLOBal Command
SET VARiable Command
SHow VARiable Command

# SHow HOst Command

## Description

**SHow HOst** displays all remote hosts currently connected to the local host in this NetEx/eFT session. Connections are established with the CONNECT command. The list displayed by **SHow HOst** includes a host connection number, the host name, logged in username, and which host (if any) is the current "active" remote host. The host number that is displayed can be used as input to the **SET HOst** command.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| SHow HOst | | |

Where:

**SHow** (required) the verb for this command. The minimum spelling is SH.

**HOst** (required) the subject for this command. The minimum spelling is HO.

## Examples

Assume the following connections have already been made:

```
eFT> connect vaxa scott john
eFT> connect sun01 meyers ed
```

Typing the **SHow HOst** command then would result in:

```
eFT> show host
eFT:
eFT:             (1) Host=vaxa          User=scott
eFT: active --> (2) Host=sun01         User=meyers
eFT:
```

## Related Topics

CONnect Command
DISconnect Command
SET HOst Command

# SHow QUALifier Command

## Description

**SHow QUALifier** lists the valid qualifiers for the specified NetEx/eFT command and gives a brief definition of each. Note that some commands require a remote connection to show a complete list of qualifiers with the **SHow QUALifier** command. These commands, for example RECEIVE, gather information from the remote host and may display only a partial list without a connection. They also may just return an error message.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| SHow QUALifier | | command |

Where:

**SHow** (required) the verb for this command. The minimum spelling is SH.

**QUAlifier** (required) the subject for this command. The minimum spelling is QUA.

**command** (required) the name of a NetEx/eFT command that supports the use of command qualifiers (e.g., SEND, LOCAL, etc.).

## Examples

To list all of the valid qualifiers for the INPUT command you would type:

```
eFT> show qualifier input
eFT: CONTinue .. continue on error (on/off)
eFT: ECHO ..... echo input to terminal (on/off)
eFT: PROMpt .... prompt string for USER input
eFT: PROMPT2 ... secondary prompt for input continuation
eFT: SEARCH .... search path for default INPUT commands
eFT: VERify .... verify string substitution (on/off)
```

## Related Topics

SHOW Command
SET Command

# SHow VARiable Command

## Description

The **SHow VARiable** command displays the currently assigned value of the variable name specified. If no variable name is specified then all variables and their values are displayed.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| SHow VARiable | | name |

Where:

**SHow** (required) the verb for this command. The minimum spelling is SH.

**VARiable** (required) the keyword for this command. The minimum spelling is VAR.

**name** (optional) variable name. NetEx/eFT variable names must be alphanumeric and no longer than 20 characters.

## Examples

The following commands define two variables, “first” and “last”:

```
eFT> set variable first john
eFT> set variable last doe
```

The following **SHow VARiable** commands display the values of the variables “first” and “last”:

```
eFT> show variable first
eFT:
eFT: FIRST ............. john
eFT:
eFT> show variable
eFT:
eFT: FIRST ............. john
eFT: LAST .............. doe
eFT:
```

## Related Topics

SET VARiable Command
SET GLOBal Command
SHow GLOBal Command

# TEXt Command

## Description

Command **TEXt** writes a string of text to the user's terminal and/or output file. **TEXt** is usually used within NetEx/eFT input scripts or aliases for interaction with a user. The string parameter may contain string substitution syntax.

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| TEXt | | [string] |

Where:

**TEXt** (required) the verb for this command. The minimum spelling is TEX.

**string** (optional) a string of text to be written out to the terminal or output file. The string may contain string substitution syntax such as string functions and references to variables.

## Examples

To display a simple line of text on the screen:

```
eFT> text this is a line of text to echo
eFT: this is a line of text to echo
```

You can use **TEXt** with a string containing string substitution syntax. For example, assuming you have a NetEx/eFT variable called "NAME" defined to be "Paul", you can display its value within a text string as:

```
eFT> text Is your name {NAME}?
eFT: Is your name Paul?
```

## Related Topics

ASK Command
INput Command

# TRanslate Command

## Description

The **TRanslate** command is used to specify and display NetEx/eFT code conversion tables, to enable and disable NetEx/eFT code conversion, and to indicate whether or not NetEx/eFT code conversion is currently enabled.

The **TRanslate** command followed by an action lets the user display and control the use of the NetEx/eFT translation tables. In NetEx/IP environments, NETEX code conversion is done by default unless the user explicitly turns on NetEx/eFT translation with the "**TRanslate ON**" command. NetEx/eFT translation can be turned off (i.e., NETEX code conversion is enabled) with the command "**TRanslate OFF**". The NetEx/eFT translation tables can be displayed using the **TRanslate** command followed by the actions **DIsplay** or **FUll**. The **REset** action reinitializes the NetEx/eFT translation tables.

The **TRanslate** command is also used to tailor the default NetEx/eFT translation tables. The initial NetEx/eFT translation tables are identical to the NETEX translation tables (in NetEx/IP environments) until modifications are made to the NetEx/eFT tables by using the **TRanslate** command followed by a native character code and a remote character code.

The initial invocation of **TRanslate** (that is, any **TRanslate** command) loads the NetEx/eFT tables with the defaults and then uses the SEARCH qualifier to make changes to the NetEx/eFT translation tables. The "**TRanslate Reset**" command reinitializes the NetEx/eFT tables with the defaults and uses the **SEARCH** qualifier again, to make changes to the NetEx/eFT tables.

When NetEx/eFT translation is enabled with the command "**TRanslate ON**", the NetEx/eFT tables are used to convert native character codes into remote character codes, and remote character code into native character codes.

The qualifiers **IN_only** and **OUT_only** allow the tables to be modified in one direction. By default, modifications are made to both the incoming and outgoing tables. For example, if the user requests that an incoming EBCDIC cent sign be converted to an ASCII7 left bracket, then an outgoing ASCII7 left bracket is converted into an EBCDIC cent sign. By specifying either **IN_only** or **OUT_only**, only the incoming or outgoing table is modified.

The **TRanslate** command without any arguments displays the status of NetEx/eFT translation, either enabled or disabled.

The **TRanslate** command does not allow the user to change the following native characters since these characters are required for NetEx/eFT protocol:

upper case alphabetic (A-Z)
digits (0-9)
space
equal sign (=)
null

## Format

| Command | Qualifier | Parameter |
|---|---|---|
| TRanslate | [-IN_only \| OFF \| ]<br>\| ON \| <br>[-OUT_only \| OFF \| ]<br>\| ON \| <br>[-SEArch string] | \| [native] [remote] [comment] \|<br>\| [action] \| |

Where:

**TRanslate** (required) is the keyword for this command. The minimum spelling is TR.

**-IN_only** (optional) is used to modify only the incoming NetEx/eFT code conversion table. The default is OFF. The minimum spelling is -IN.

**-OUT_only** (optional) is used to modify only the outgoing NetEx/eFT code conversion table. The default is OFF. The minimum spelling is -OUT.

**-SEArch** (optional) the "**TRanslate SEArch**" path is used to find default translation tables for the different HOSTCODES. On an IBM z/OS system the default search path is "(SITE)". Each time a translate command is issued NetEx/eFT looks in the "SITE" partitioned data set for the member "{HOSTCODE:remote}". For additional information, refer to "IBM z/OS NetEx/eFT SEARCH Keywords (SITE), (USER), and (NONE)" on page 106. The minimum spelling is -SEA.

**action** (optional) describes the action **TRanslate** takes. The action can be one of the following:

| | |
|---|---|
| **DIsplay** | display differences from NETEX tables |
| **FUll** | display entire translate table |
| **OFF** | disable NetEx/eFT translation |
| **ON** | enable NetEx/eFT translation |
| **REset** | reset table and process search path |

**native** (optional) the native character code (in octal, decimal, or hexadecimal format).

**remote** (optional) the remote character code (in octal, decimal, or hexadecimal format).

**comment** (optional) descriptive comment.

## Examples

### Example #1:

Suppose the native character set is EBCDIC and the remote host has a HOSTCODE value of ASCII7. A translation table is setup to translate the EBCDIC cent sign (0x4A) and solid bar (0x4F) (which are invalid ASCII7 characters) to the ASCII7 left bracket (0x5B) and right bracket (0x5D), the following NetEx/eFT commands would update the default translation tables:

```
eFT> translate 0x4A 0x5B (cent sign <--> left bracket)
eFT> translate 0x5D 0x4D (solid bar <--> right bracket)
```

NetEx/eFT translation is enabled with the following command:

```
eFT> translate on
```

This translation effects both directions. The EBCDIC cent sign (0x4A) is converted into a ASCII7 left bracket (0x5B) on its way out, and the ASCII7 left bracket (0x5B) is converted into and EBCDIC cent sign (0x4A) on its way in.

### Example #2:

Suppose the native character set is EBCDIC and the remote host has a HOSTCODE value of ASCII7:

```
eFT> connect VAX user pw -quiet
```

The current "**TRanslate SEArch**" path is:

```
eFT> show translate search
eFT:
eFT: SEArch . . . . . .  (SITE)
eFT:
```

NetEx/eFT translation is turned on with the following command:

```
eFT> translate on
```

At this point NetEx/eFT uses the **SEArch** path to look for a NetEx/eFT script file using the remote HOSTCODE, in this case ASCII7. For an IBM z/OS system the **SEArch** path tells NetEx/eFT to look for the partitioned data set member "ASCII7" in the "(SITE)" location. The partitioned data set member "ASCII7" could contain the lines from the above example:

```
* Sample ASCII7 translation table for
* an EBCDIC host
*
translate 0x4A 0x5B (cent sign <--> left bracket)
translate 0x4F 0x5D (solid bar <--> right bracket)
```

Now every time a connection is active to an ASCII7 host and the NetEx/eFT translation tables are initialized, the table is automatically loaded.

The translate table can be displayed with the command:

```
eFT> translate full
```

### Example #3:

The **TRanslate** search path allows the user to select optional files for input when the **TRanslate** command is issued. Suppose instead of loading the default tables for the active host, "{HOSTCODE:REMOTE}", the user would rather load a table from the user's default TSOPREFIX. The current search path is:

```
eFT> show translate search
eFT:
eFT: SEArch ............. (SITE)
eFT:
```

The search path can be changed with the command:

```
eFT> set translate search (USER)translate.ua
eFT> show translate search
eFT:
eFT: SEArch ............. (USER)translate.ua
eFT:
```

The search path for the **TRanslate** command now implies, look for the Sequential Data Set “translate.ua” in the default TSOPREFIX. The following command turns on translation and initializes the translate tables by reading the sequential data set “TRANSLATE.UA” in the user’s default TSOPREFIX:

```
eFT> translate on
```

# Host Independent Commands

This section contains descriptions of the built-in host independent commands. In Table 4, the first column lists the z/OS host-independent command (defined as a local alias command), and the second column lists the corresponding z/OS system command that gets executed whenever the local alias command is issued.

Refer to the "Issuing Local IBM z/OS Host-Independent Commands" section on page 30 for a description of the command that is executed.

**Table 4. z/OS Host Independent Commands**

| z/OS Host Independent Command | z/OS System Command |
|---|---|
| CANcel | CANCEL |
| COPY | SMCOPY FROMDATASET({1}) TODATASET({2}) NOTRANS |
| DELete | DELETE |
| DIFference | (TYPE)DIFFERENCE is not available for MVS |
| DIRectory | LISTCAT |
| HELP | HELP |
| PRInt | PRINTDS DATASET({1}) {2} {3} {4} {5} |
| QUEue | STATUS |
| RENname | RENAME |
| SUBmit | SUBMIT |
| TYPe | %EFTTYPE |
| WHO | (TYPE)WHO is not available for MVS |

# General Alias Commands

This section contains descriptions of the following alias commands:

ASsign
EDit
LDir
LEDit
LOGIn
LOGOn
RDir
RECPRT
SET LOgin
SHOW LOgin
SLD
SRD
TESt
TSO

**Note:** There are three special aliases that will be displayed in the output of the **SHOW ALIAS** command: **ISPFMENU**, **NUAICMD**, and **NUAISPF**. These aliases are not meant to be called from a NetEx/eFT script or from the NetEx/eFT client command line. They exist solely to facilitate the eFT213 ISPF panels feature.

# ASsign Alias Command

## Description

Assign the results of a string function to the named variable. Function parameters are separated by blanks. Enclose string literal parameters in double quotes.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| `ASsign` | | `variable function [parameters...]` |

Where:

**ASsign** (required) the verb for this command. The minimum spelling is AS.

**variable** (required) the name of a variable.

**function** (required) the string function to be applied.

**parameters** (optional) additional parameters for the string function.

## Examples

```
eFT> assign sum add 1 2
eFT> show variable sum
eFT: SUM ............... 3
```

# DEBUGOFF Alias Command & DBGOFF script

## Description

An alias that turns off debug parameters from a single command. This command should be used specifically under the direction of NetEx Support personnel. This will stop the production of the debug messages (for the client only) which were turned on by the **DEBUGON** alias command.

This alias and script are only to be used under direction from NESi Support.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| DEBUGOFF | | |

Where:

**DEBUGOFF** (required) the verb for this command. The minimum spelling is DEBUGOFF. The Debug settings will resemble the following:

```
CONnect ..... off
COUnt ..... 4096
CRC ..... off
DIScard ..... off
FILe ..... off
HEX ..... off
HSM ..... off
INPut ..... off
INTerval ..... 0
LOCal ..... off
LOG ..... off
MEMory ..... off
MESsages ..... off
MVS_DIAGNS ..... off
PARse ..... off
PROTocol ..... off
RECeive ..... off
REMote ..... off
SENd ..... off
TCP_READ ..... off
TCP_WRITE ..... off
```

## Related Topics

DEBUGON Alias Command & DBGON script

# DEBUGON Alias Command & DBGON script

## Description

An alias for setting the debug parameters. This command should be used specifically under the direction of NetEx Support personnel. This will produce an overabundance of messages (for the client only) which can be used to aid in diagnosing problems encountered by users. (Due to the volume of output, we recommend that your 3270 emulator be in logging mode so the output can be sent to NetEx Support for analysis.)

This alias and script are only to be used under direction from NESi Support.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| DEBUGON | | |

Where:

**DEBUGON** (required) the verb for this command. The minimum spelling is DEBUGON. The Debug settings will resemble the following:

CONnect ..... on
COUnt ..... 4096
CRC ..... on
DIScard ..... off
FILe ..... on
HEX ..... off
HSM ..... off
INPut ..... on
INTerval ..... 0
LOCal ..... on
LOG ..... on
MEMory ..... off
MESsages ..... on
MVS_DIAGNS ..... on
PARse ..... on
PROTocol ..... on
RECeive ..... on
REMote ..... on
SENd ..... on
TCP_READ ..... on
TCP_WRITE ..... on

## Related Topics

DEBUGOFF Alias Command & DBGOFF script

# EDit Alias Command

## Description

Edit a remote file with the user's local full screen editor.

This alias will receive the remote file to a local temporary file named "edit.tmp". Invoke the local full screen editor (using the LEDIT alias). After the user exits the editor, issue an "Update remote?" prompt to determine if the remote file should be updated with the changes. If the response is 'yes' (the default), send the temporary file back to the remote host replacing the original file. Then delete the 'edit.tmp' file.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| EDit | | remote_filename |

Where:

**EDit** (required) the verb for this command. The minimum spelling is ED.

**remote_filename** (required) the name of the remote file to be edited.

## Examples

```
eFT> edit test.txt
```

# LDir Alias Command

## Description

A shorthand alias for the command LOCAL DIRECTORY.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| LDir | | [directory] |

Where:

**LDir** (required) the verb for this command. The minimum spelling is LD.

**directory** (optional) the name of the local directory. If no directory name is provided, the current local directory is used.

## Examples

```
eFT> ldir
NT:  Volume in drive C has no label.
NT:  Volume Serial Number is FC6C-2987
NT:
NT:  Directory of C:\
NT:
NT: 11/01/00  11:14                12,037 aoedoppl.txt
NT: 11/01/00  11:15                 1,480 aoeWVlog.txt
NT: 07/08/99  11:19                     0 AUTOEXEC.BAT
NT: 04/02/01  15:42                   291 badhand.txt
NT: 10/13/00  09:27                   288 BOOT.PQB
NT: 07/08/99  10:41         <DIR>         I386
NT: 07/08/99  11:22         <DIR>         IBM
NT: 05/23/01  12:34         <DIR>         TEMP
NT: 07/08/99  10:41         <DIR>         THINKPAD
NT: 07/08/99  11:22         <DIR>         ThinkPad 390E User's Guide
NT: 05/08/00  13:45         <DIR>         WINDOWS
NT: 11/22/00  10:03         <DIR>         Windows Update Setup Files
NT: 05/18/01  08:21         <DIR>         WINNT
NT: 05/22/01  15:03                46,565 winzip.log
NT:                5 File(s)      3,225,410 bytes
NT:                           1,031,121,920 bytes free
eFT>
```

# LEDit Alias Command

## Description

Invoke the local full screen editor for the specified ‘local_file’.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| `LEDit` | | `local_file` |

Where:

**LEDit** (required) the verb for this command. The minimum spelling is LED.

**local_file** (required) the name of the local file to be edited.

## Examples

```
eFT> ledit test.txt
```

# LOGIN Alias Command

## Description

The LOGIN alias command is used to establish a connection to a remote host on the network. The host name specified must exist in the local network hosts database (either in the local hosts file, accessible through DNS, or NetEx/IP NCT file). The LOGIN alias command issues the NetEx/eFT CONNECT command to establish the connection.

The LOGIN alias prompts for login information and issues a connect command to the specified host. Any missing parameters are prompted for. The password is captured in a "secure" mode so it does not display on the user's terminal.

Refer to the description of qualifiers in the "CONnect Command" section, on page 120, for further information.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| LOGin | | [host] [username] [password] [qualifiers...] |

Where:

**LOGIN** (required) the verb for this command. The minimum spelling is LOG.

**host** (prompt) the name of the remote host computer to which the connection should be established

**username** (prompt) the name of the user to use for the login on the remote host computer. If this parameter is not specified, the login alias will prompt the user to enter a username.

**password** (prompt) the password to use for the username login sequence on the remote host computer. If this parameter is not specified, the login alias will prompt the user to enter the password.

**qualifiers** (prompt) additional qualifiers for the CONNECT command.

## Examples

To establish a NETEX/eFT session using the LOGIN alias command, issue the following commands:

```
eFT> login
Hostname? yellowstone
Username? test1
Password? ********
Qualifiers?

[...]

yellowstone>
```

## Related Topics

FTP Alias Command
OPEN Alias Command
CONnect Command

# RDir Alias Command

## Description

Shorthand alias for REMOTE DIRECTORY.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| RDir | | [directory] |

Where:

**RDir** (required) the verb for this command. The minimum spelling is RD.

**directory** (optional) the name of the remote directory. If this argument does not exist, the current remote directory is used.

## Examples

```
yellowstone> rdir
Unix: total 658
Unix: drwxrwxrwt   7 sys      sys          410 May 17 03:30 .
Unix: drwxr-xr-x  26 root     root        1024 May  8 10:12 ..
Unix: drwxrwxr-x   2 root     root         104 May  8 10:12 .X11-pipe
Unix: drwxrwxr-x   2 root     root         104 May  8 10:12 .X11-unix
Unix: drwxrwxrwx   2 root     root         107 May  8 10:12 .pcmcia
Unix: drwxrwxrwx   2 root     other         69 May  8 10:14 .removable
Unix: drwxrwxrwt   2 root     root         207 May  8 10:11 .rpc_door
Unix: -rw-rw-r--   1 root     sys         5584 May  8 10:11 ps_data
yellowstone>
```

# RECPRT Alias Command

## Description

The RECPRT alias will receive a remote file and send it to the printer spool.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| RECPRT | | [filename] [class] [destination] |

Where:

**RECPRT** (required) the verb for this command.

**filename** (optional) the name of the remote file to print. If this argument is not specified, the alias will prompt for it.

**class** (optional) SYSOUT class to use for the print output. If a SYSOUT CLASS is not specified when this alias is called, the default class used is "a".

**destination** (optional) SYSOUT destination to use for the print output.

## Examples

```
yellowst> recprt
Remote file name? tmp.sh
11:49:01 MVS:Source                       Destination                  Size
11:49:01 MVS:-------------------------    -------------------------    ------
11:49:01 MVS:/mnt/home2/test/tmp.sh       TEST.TEST.TSU07753.D0000       345
11:49:01 MVS:                                103.?
yellowst>
```

# SET LOgin Alias Command

## Description

An alias for the SET CONNECT command.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| `SET LOgin` | | `qualifier [value]` |

Where:

**SET LOgin** (required) the verb for this command. The minimum spelling is SET LO.

**qualifier** (required) the CONNECT command qualifier to modify.

**value** (optional) the value for the CONNECT qualifier, if necessary.

## Examples

```
zpdt2> show login
eFT:
eFT:   ACCount ...........
eFT:   ADAPTer ........... NETA
eFT:   APPlication .......
eFT:   BLOCKsize ......... 16384
eFT:   CERTDB ............ /users/huhned/test.kdb
eFT:   CERTLB ............ zos5r
eFT:   CERTPW ............ testtest
eFT:   COMmand ...........
eFT:   INTerval .......... 5
eFT:   PASSword ..........
eFT:   PROFile ...........
eFT:   PROJect ...........
eFT:   QUIet ............. off
eFT:   SCRIpt ............
eFT:   SEArch ............ (SITE) (USER)
eFT:   SECondary .........
eFT:   SECure ............ off
eFT:   SERvice ........... EFT2
eFT:   SITE ..............
eFT:   TIMeout ........... 33
eFT:   USERname ..........
eFT:   VERBose ........... on
eFT:
zpdt2>
```

```
eFT> set login quiet off
eFT> show login
eFT:
eFT:   ACCount ..........
eFT:   APPlication .......
eFT:   BLOCKsize ......... 16384
eFT:   COMmand ..........
eFT:   FULL .............. off
eFT:   INTerval .......... 5
eFT:   NODE ..............
eFT:   PASSword ..........
eFT:   PROFile ...........
eFT:   PROJect ...........
eFT:   QUIet ............. off
eFT:   SCRIpt ............
eFT:   SEArch ............ (SITE) (USER)
eFT:   SECondary .........
eFT:   SERvice ........... USER
eFT:   SITE ..............
eFT:   TIMeout ........... 120
eFT:   USERname ..........
eFT:   VERBose ........... on
eFT:
yellowstone>
```

## Related Topics

CONnect Command
SET Command

# SHow LOgin Alias Command

## Description

An alias for the SHOW CONNECT command.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| SHow LOgin | | |

Where:

**SHow LOgin** (required) the verb for this command. The minimum spelling is SH LO.

## Examples

```
zpdt2> show login
eFT:
eFT:   ACCount ...........
eFT:   ADAPTer ........... NETA
eFT:   APPlication .......
eFT:   BLOCKsize ......... 16384
eFT:   CERTDB ............ /users/huhned/test.kdb
eFT:   CERTLB ............ zos5r
eFT:   CERTPW ............ testtest
eFT:   COMmand ...........
eFT:   INTerval .......... 5
eFT:   PASSword ..........
eFT:   PROFile ...........
eFT:   PROJect ...........
eFT:   QUIet ............. off
eFT:   SCRIpt ............
eFT:   SEArch ............ (SITE) (USER)
eFT:   SECondary .........
eFT:   SECure ............ off
eFT:   SERvice ........... EFT2
eFT:   SITE ..............
eFT:   TIMeout ........... 33
eFT:   USERname ..........
eFT:   VERBose ........... on
eFT:
zpdt2>
```

## Related Topics

CONnect Command
SHOW Command

# SLD Alias Command

## Description

A shorthand alias for the SET LOCAL DIRECTORY command.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| SLD | | [directory] |

Where:

**SLD** (required) the verb for this command. The minimum spelling is SLD.

**directory** (optional) the name of the local directory.

## Examples

```
eFT> show local dir
eFT: DIRectory ......... C:\
eFT> sld \temp
eFT> show local dir
eFT: DIRectory ......... C:\TEMP
eFT>
```

## Related Topics

SET ALias Command
SHow ALias Command

# SRD Alias Command

## Description

A shorthand alias for SET REMOTE DIRECTORY.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| SRD | | [directory] |

Where:

**SRD** (required) the verb for this command. The minimum spelling is SRD.

**directory** (optional) the name of the remote directory.

## Examples

```
yellowstone> show remote dir
14:26:07 MVS:DIRectory ......... /tmp
yellowstone> srd /tmp
yellowstone> show remote dir
14:26:19 MVS:DIRectory ......... /tmp
yellowstone>
```

## Related Topics

SET ALias Command
SHow ALias Command

# TESt Alias Command

## Description

The TEST alias can be used with any of the numeric compare functions (EQ, NE, LE, LT, GE, GT) and the string compare functions (EQS, NES, CMP). All of these functions require two parameters. Function parameters CANNOT contain embedded blanks. String literal parameters are enclosed in double quotes.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| TESt | | param1 function param2 action |

Where:

**TESt** (required) the verb for this command. The minimum spelling is TES.

**param1** (required) a string function, string variable, or string literal.

**function** (required) the string function to be performed.

**param2** (required) a string function, string variable, or string literal.

**action** (required) the action to be taken. A NetEx/eFT command or alias.

## Examples

If the current year is “18”, print a text message:

```
eFT> TEST EXT(DATE(1),1,2) EQ 18 TEXT Welcome to 2019
13:10:06 MVS:Welcome to 2019
```

# TSO Alias Command

## Description

The TSO alias command can be used as a shorthand for the eFT “LOCal Command”.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| TSO | | command |

Where:

**TSO** (required) the verb for this command.

**command** (required) a local system command.

# FTP Alias Commands

This section contains descriptions of the following FTP alias commands:

ACCOUNT
APPEND
ASCII
BIN
BYE
CD
CLOSE
DELETE
DIR
FTP
GET
LCD
LS
LSMem
MKDIR
OPEN
PUT
PWD
RENAME
RM
RMDIR

# ACCOUNT Alias Command

## Description

The **ACCOUNT** alias command issues **ACCOUNT** as a system command on the remote host computer to which there is an active connection. Command availability is dependent on remote host type.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| ACCOUNT | | [string] |

Where:

**ACCOUNT** (required) the verb for this command. The minimum spelling is ACCOUNT.

**string** (optional) a string of text to be passed as input to the remote ACCOUNT system com-mand. The format of the parameters for this command is dependent on the type of the remote host system. Refer to the remote user's guide section of the NetEx/eFT manual for the remote system to which the command is being issued.

## Examples

To issue the **ACCOUNT** command on the remote system:

```
eFT> account
```

## Related Topics

FTP Alias Command
LOGIN Alias Command
CONnect Command

# APPEND Alias Command

## Description

The **APPEND** alias command sends the source file from the local host to the current active remote host and appends it to the destination file. If no path to the file is specified on the source or destination file (i.e., if a file name is given without a directory or device specification), the default local and remote directories are used respectively. That is, the source file is assumed to exist in the local default directory, and the appended file is assumed to exist in the remote default directory. If the destination parameter is not specified, a file by the same name as the source file name will be appended to in the remote default directory. If the remote file does not exist, the command will return an error.

The source file name may include the NetEx/eFT wildcard characters "*" and "?", as well as host specific wildcard characters where the two do not conflict. See "Source Wildcard Support for IBM z/OS File Transfers" on page 56 and "Destination Wildcard Support for IBM z/OS File Transfers" on page 58 for more details.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| APPEND | | source [destination] |

Where:

**APPEND** (required) the verb for this command. The minimum spelling is APPEND.

**source** (required) the file specification for the file on the local host that you intend to send to the remote host.

**destination** (optional) the file specification for the file that is being appended to on the remote host.

## Examples

To append the file "alpha.txt" from the current default local directory ("C:\GUEST\SMITH\") to a file on the remote host, issue the following command:

```
eFT> append alpha.txt
eFT: SOURCE                      DESTINATION           SIZE
eFT: ------------------------  -----------------  -------
eFT: C:\GUEST\SMITH\ALPHA.TXT   ALPHA.TXT            54909
```

Notice the entire source filename is displayed. The resulting destination file specification depends on the remote host to which the connection is made. If no destination name is specified, the source name is used to construct the destination name of a file in the current default remote directory. The size indicated in the display represents an approximation of the number of bytes from the source file transferred.

## Related Topics

FTP Alias Command
SENd Command

# ASCII Alias Command

## Description

The **ASCII** alias command sets the default **SEND** and **RECEIVE** transfer mode to **CHARACTER**.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| ASCII | | |

Where:

**ASCII** (required) the verb for this command. The minimum spelling is ASCII.

There are no qualifiers or parameters with this command.

## Examples

To receive the ASCII file “alpha.txt” from host “NT2” using the FTP aliases, issue the following commands:

```
eFT> ftp nt2
User: smith
Password: ********
eFT> ascii
eFT> get alpha.txt
eFT: SOURCE                    DESTINATION          SIZE
eFT: ------------------------  -----------------    -------
eFT: C:\GUEST\ALPHA.TXT        C:\SMITH\ALPHA.TXT   54909
eFT> bye
```

## Related Topics

FTP Alias Command
GET Alias Command
PUT Alias Command
APPEND Alias Command
SENd Command
RECeive Command

# BIN Alias Command

## Description

The **BIN** alias command sets the default **SEND** and **RECEIVE** transfer mode to **STREAM**.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| BIN | | |

Where:

**BIN** (required) the verb for this command. The minimum spelling is BIN.

There are no qualifiers or parameters with this command.

## Examples

To send the BINARY file “test.exe” to host “NT3” using the FTP aliases, issue the following commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> bin
eFT> put test.exe
eFT: SOURCE                    DESTINATION          SIZE
eFT: ------------------------  -----------------    -------
eFT: C:\GUEST\TEST.EXE         C:\SMITH\TEST.EXE    193000
eFT> bye
```

## Related Topics

FTP Alias Command
GET Alias Command
PUT Alias Command
APPEND Alias Command
SENd Command
RECeive Command

# BYE Alias Command

## Description

The **BYE** alias command causes NetEx/eFT to terminate.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| BYE | | |

Where:

**BYE** (required) the verb for this command. The minimum spelling is BYE.

There are no qualifiers or parameters with this command.

## Examples

To terminate NetEx/eFT by using the FTP aliases, issue the **BYE** command:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> bye
```

## Related Topics

FTP Alias Command
EXit Command

# CD Alias Command

## Description

The **CD** alias command sets the default remote directory on the remote host computer to which there is an active connection, to the directory name specified by the command.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| CD | | [directory] |

Where:

**CD** (required) the verb for this command. The minimum spelling is CD.

**directory** (optional) the name of the remote directory to set as the default.

**Note:** the exact format of the **CD** command is dependent on the type of the remote host computer (e.g., MVS, Unix, NT, etc).

## Examples

During a NetEx/eFT session to host “NT3”, to set the default remote directory to “C:\JONES\TEST”, issue the following FTP alias commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> cd C:\JONES\TEST
```

## Related Topics

FTP Alias Command
LCD Alias Command

# CLOSE Alias Command

## Description

The **CLOSE** alias command terminates the connection from the remote host computer to which there is an active connection.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| CLOSE | | |

Where:

**CLOSE** (required) the verb for this command. The minimum spelling is CLOSE.

There are no qualifiers or parameters with this command.

## Examples

To terminate a NetEx/eFT session using the FTP aliases, issue the **CLOSE** command:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> ftp nt4
User: smith
Password: ********
eFT> close              (note: terminates connection to nt4)
eFT> set host 1         (sets host connection back to first)
eFT> close              (note: terminates connection to nt3)
```

## Related Topics

FTP Alias Command
DISconnect Command

# DELETE Alias Command

## Description

The **DELETE** alias command issues the system **DELETE** command on the remote host to which there is an active connection. The filename specified by the command is deleted.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| DELETE | | filename |

Where:

**DELETE** (required) the verb for this command. The minimum spelling is DELETE.

**filename** (required) the name of the file on the remote host computer to delete.

**Note:** the exact format of the **DELETE** command is dependent on the type of the remote host computer (e.g., MVS, Unix, NT, etc).

## Examples

During a NetEx/eFT session to host "NT3", to delete the file "test.exe" in directory "C:\JONES", issue the following FTP alias commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> cd C:\JONES
eFT> delete test.exe
eFT> bye
```

## Related Topics

FTP Alias Command
CD Alias Command

# DIR Alias Command

## Description

The **DIR** alias command issues **DIRECTORY** as a system command on the remote host computer to which there is an active connection. The contents of the directory specified by the command are displayed.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| DIR | | [directory] |

Where:

**DIR** (required) the verb for this command. The minimum spelling is DIR.

**directory** (optional) the name of the directory to display. If not specified, the current working directory is displayed.

**Note:** the exact format of the **DIR** command is dependent on the type of the remote host computer (e.g., MVS, Unix, NT, etc).

## Examples

During a NetEx/eFT session to host "NT3", to display the contents of directory "C:\JONES", issue the following FTP alias commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> dir c:\jones
eFT> bye
```

## Related Topics

FTP Alias Command
CD Alias Command

# FTP Alias Command

## Description

The FTP alias command is used to establish a connection to a remote host on the network. The host name specified must exist in the local TCP/IP "hosts" database (either in the local hosts file, or accessible through DNS). The FTP alias command issues the NetEx/eFT CONNECT command to establish the connection. Refer to the "CONnect Command" on page 120 for a description of CONNECT processing.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| FTP | | host [username] [password] |

Where:

**FTP** (required) the verb for this command. The minimum spelling is FTP.

**host** (required) the name of the remote host computer to which the connection should be established

**username** (required) the name of the user to use for the login on the remote host computer. If this parameter is not specified, the ftp alias will prompt the user to enter a username.

**password** (required) the password to use for the username login sequence on the remote host computer. If this parameter is not specified, the ftp alias will prompt the user to enter the password.

## Examples

To establish a NetEx/eFT session using the **FTP** alias commands, issue the following commands:

```
eFT> ftp nt2
User: smith
Password: ********
eFT>
```

## Related Topics

LOGIN Alias Command
OPEN Alias Command
CONnect Command

# GET Alias Command

## Description

The **GET** alias command receives the source file from the current active remote host and saves it as a destination file on the local host. If no path to the file is specified on the source or destination file (i.e., if a file name is given without a directory or device specification), the default remote and local directories are used respectively. That is, the source file is assumed to exist in the remote default directory, and the newly received file will be created in the local default directory. If the destination parameter is not specified at all, a file by the same name as the source file name will be created in the local default directory.

The source file name may include the NetEx/eFT wildcard characters "*****" and "**?**", as well as host specific wildcard characters where the two do not conflict. See the discussion on NetEx/eFT wildcarding in "Source Wildcard Support for IBM z/OS File Transfers" on page 56 for further details.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| GET | | source [destination] |

Where:

**GET** (required) the verb for this command. The minimum spelling is GET.

**source** (required) the file specification for the file on the remote host that you intend to receive.

**destination** (optional) the file specification for the new file that is to be created on the local host.

## Examples

To receive the file "alpha.txt" from host "NT2" using the **FTP** aliases, issue the following commands:

```
eFT> ftp nt2
User: smith
Password: ********
eFT> get alpha.txt
eFT: SOURCE                    DESTINATION          SIZE
eFT: ------------------------  -----------------    -------
eFT: C:\GUEST\ALPHA.TXT        C:\SMITH\ALPHA.TXT   54909
eFT> bye
```

## Related Topics

FTP Alias Command
PUT Alias Command
RECeive Command

# LCD Alias Command

## Description

The **LCD** alias command sets the default local directory on the local host computer to the name specified by the command.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| LCD | | directory |

Where:

**LCD** (required) the verb for this command. The minimum spelling is LCD.

**directory** (required) the name of the local directory to set as the default.

## Examples

During a NetEx/eFT session to host "NT4", set the default local directory to "C:\SMITH\TEST" by issuing the following FTP alias commands:

```
eFT> ftp nt4
User: smith
Password: ********
eFT> lcd C:\SMITH\TEST
```

## Related Topic

FTP Alias Command
CD Alias Command

# LS Alias Command

## Description

The "LS" alias command issues DIRECTORY as a system command on the remote host computer to which there is an active connection. The contents of the directory specified by the command are displayed.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| LS | | [directory] |

Where:

**LS** (required) the verb for this command. The minimum spelling is LS.

**directory** (optional) the name of the directory to list.

**Note:** the exact format of the **LS** command is dependent on the type of the remote host computer (e.g., MVS, Unix, NT, etc).

## Examples

During a NetEx/eFT session to host "NT3", issue the following FTP alias commands to list the contents of directory "C:\JONES":

```
eFT> ftp nt3
User: smith
Password: ********
eFT> cd c:\jones
eFT> ls
eFT> bye
```

## Related Topics

DIR Alias Command
CD Alias Command
LCD Alias Command

# LSMem Alias Command

## Description

The **LSMem** alias command issues a DIRECTORY command on the remote host computer to which there is an active connection. The contents of the directory specified by the command are displayed.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| `LSMem` | | `[directory]` |

Where:

**LSMem** (required) the verb for this command. The minimum spelling is LSM.

**directory** (optional) the name of the directory to display.

**Note:** the exact format of the **LSMEM** command is dependent on the type of the remote host computer (e.g., MVS, Unix, NT, etc).

## Examples

During a NetEx/eFT session to host "NT3", issue the following FTP alias commands to list the contents of directory "C:\JONES\TEST":

```
eFT> ftp nt3
User: smith
Password: ********
eFT> lsm c:\jones\test
eFT> bye
```

## Related Topics

DIR Alias Command
CD Alias Command
LCD Alias Command

# MKDIR Alias Command

## Description

The **MKDIR** alias command issues the "mkdir" command on the remote host computer to which there is an active connection. A new directory is created in the current working directory, having a name specified by the command.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| MKDIR | | directory |

Where:

**MKDIR** (required) the verb for this command. The minimum spelling is MKDIR.

**directory** (required) the name of the directory to create.

**Note:** the exact format of the **MKDIR** command is dependent on the type of the remote host computer (e.g., MVS, Unix, NT, etc.).

## Examples

During a NetEx/eFT session to host "NT3", issue the following FTP alias commands to create a new directory in "C:\JONES" called "TEST":

```
eFT> ftp nt3
User: smith
Password: ********
eFT> cd C:\JONES
eFT> mkdir test
eFT> bye
```

## Related Topics

DIR Alias Command
CD Alias Command
LCD Alias Command

# OPEN Alias Command

## Description

The **OPEN** alias command is used to establish a connection to a remote host on the network. The host name specified must exist in the local TCP/IP "hosts" database (either in the local hosts file, or accessible through DNS). The **OPEN** alias command issues the NetEx/eFT **CONNECT** command to establish the connection. Refer to the "CONnect Command" on page 120 for a description of **CONNECT** processing.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| OPEN | | [host] [username] [password] |

Where:

**OPEN** (required) the verb for this command. The minimum spelling is OPEN.

**host** (required) the name of the remote host computer to which the connection should be established

**username** (required) the name of the user to use for the login on the remote host computer. If this parameter is not specified, the ftp alias will prompt the user to enter a username.

**password** (required) the password to use for the username login sequence on the remote host computer. If this parameter is not specified, the ftp alias will prompt the user to enter the password.

## Examples

To establish a NetEx/eFT session using the FTP alias commands, issue the following commands:

```
eFT> ftp
eFT> open
Hostname: nt3
User: smith
Password: ********
eFT>
```

To specify parameters on the **OPEN** command line, issue the following commands:

```
eFT> ftp
eFT> open nt3 smith
Password: ********
eFT>
```

## Related Topics

LOGIN Alias Command
FTP Alias Command
CONnect Command

# PUT Alias Command

## Description

The **PUT** alias command sends the source file from the local host to the current active remote host and saves it as a destination file. If no path to the file is specified for the source or destination file (i.e., if a file name is given without a directory or device specification), the default local and remote directories are used respectively. That is, the source file is assumed to exist in the local default directory, and the new file is created in the remote default directory. If the destination parameter is not specified at all, a file by the same name as the source file name will be created in the remote default directory.

The source file name may include the NetEx/eFT wildcard characters "*****" and "**?**", as well as host specific wildcard characters where the two do not conflict. See "Source Wildcard Support for IBM z/OS File Transfers" on page 56 and "File Handling Under IBM z/OS NetEx/eFT" on page 41 for more details.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| PUT | | source [destination] |

Where:

**PUT** (required) the verb for this command. The minimum spelling is PUT.

**source** (required) the file specification for the file on the local host that you intend to send.

**destination** (optional) the file specification for the file on the remote host.

## Examples

To send the file "alpha.txt" from host "NT2" to host "NT3" using the FTP aliases, issue the following commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> put alpha.txt
eFT: SOURCE                    DESTINATION          SIZE
eFT: ------------------------  -----------------    -------
eFT: C:\GUEST\ALPHA.TXT        C:\SMITH\ALPHA.TXT   54909
eFT> bye
```

## Related Topics

FTP Alias Command
GET Alias Command
SENd Command

# PWD Alias Command

## Description

The **PWD** alias command issues the **SHOW REMOTE DIRECTORY** command on the remote host computer to which there is an active connection. On remote hosts that support this command, the path to the current working directory is displayed.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| PWD | | |

Where:

**PWD** (required) the verb for this command. The minimum spelling is PWD.

There are no qualifiers or parameters with this command.

## Examples

To establish a connection to remote host "NT3", and display the current path information, issue the following FTP alias commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> pwd
eFT>
```

## Related Topics

FTP Alias Command
CD Alias Command

# RENAME Alias Command

## Description

The **RENAME** alias command issues the RENAME system command on the remote host computer to which there is an active connection. The file specified by "file1" is renamed to "file2".

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| RENAME | | file1 file2 |

Where:

**RENAME** (required) the verb for this command. The minimum spelling is RENAME.

**file1** (required) the name of an existing file to rename.

**file2** (required) the new name for the file.

**Note:** The exact format of the **RENAME** command is dependent on the type of the remote host computer (e.g., MVS, Unix, NT, etc).

## Examples

To establish a connection to remote host "NT3" and rename "alpha.txt" to "test.txt", issue the following FTP alias commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> rename alpha.txt test.txt
eFT>
```

## Related Topics

FTP Alias Command

# RM Alias Command

## Description

The **RM** alias command issues the DELETE system command on the remote host computer to which there is an active connection. The filename specified by the command is deleted.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| RM | | filename |

Where:

**RM** (required) the verb for this command. The minimum spelling is RM.

**filename** (required) the name of the file to delete.

**Note:** The exact format of the RM command is dependent on the type of the remote host computer (e.g. MVS, Unix, NT, etc).

## Examples

To establish a connection to remote host “NT3” and delete “test.txt”, issue the following FTP alias commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> rm test.txt
eFT>
```

## Related Topics

FTP Alias Command

# RMDir Alias Command

## Description

The **RMDIR** alias command issues the RMDIR system command on the remote host computer to which there is an active connection. The directory specified by the command is deleted.

## Format

| Command | Qualifiers | Parameters |
|---|---|---|
| RMDir | | [directory] |

Where:

**RMDir** (required) the verb for this command. The minimum spelling is RMD.

**directory** (required) the name of the directory to delete.

**Note:** The exact format of the RM command is dependent on the type of the remote host computer (e.g. MVS, Unix, NT, etc).

## Examples

To establish a connection to remote host "NT3" and delete the "alpha" directory, issue the following FTP alias commands:

```
eFT> ftp nt3
User: smith
Password: ********
eFT> rmdir alpha
eFT>
```

## Related Topics

FTP Alias Command
MKDIR Alias Command

# Appendix A. NetEx/eFT Encrypt Alias

## Overview

The Netex/eFT **Encrypt** alias can be used to encrypt user passwords which will be used by NetEx/eFT to establish connections to remote hosts. All other data transmission will be in plain text.

Sites often store remote system access information in files to be included on the **CONNECT** command during NetEx/eFT execution. To avoid storing cleartext passwords in files, you can encrypt a password and store the encrypted form of the password in the file. The encrypted password is useless to an individual to gain access to a remote system outside of NetEx/eFT. Within NetEx/eFT, the encrypted form of the password is valid only when decrypted internally by NetEx/eFT using the local host username as a secondary encryption key.

The **ENCRYPT** alias definition:

```
set alias ENCrypt {}{dfn(1, "goto skip1")} !
ask -secure -prompt "Enter password? " 1 !
skip1: {dfn(2, "goto skip2")} !
ask -prompt "Enter optional username (or press Enter)? " 2 !
skip2: set global pw {encrypt(1,2)} !
text The encrypted password is {pw}
```

## Format

| Command | Parameters |
|---|---|
| NUAENCR | password username |

Where:

**password** (STRING) specifies the remote system password you want to encrypt. The encrypted form of this password is returned by **ENCRYPT** and should be stored in a file for use with the **CONNECT** command. You may leave this parameter off the command line and be prompted for it.

**username** (STRING) specifies the local host username which represents the username associated with the local NetEx/eFT process that will issue the CONNECT command with the encrypted password. This is optional. If omitted the encrypted password can be used by anyone. This local host username is used as a secondary encryption key for the specified password. When NetEx/eFT is later run on the local system, either interactively or in batch, it queries the operating system for the username of the process. NetEx/eFT then uses this username as one of its keys in decrypting the password. The value for "username" above must be entered in uppercase in order to match the username value later returned by MVS/zOS, HPE NonStop, and OpenVMS.

## Example

Encrypting passwords stored in a NetEx/eFT input script file:

Suppose a job running under the local MVS username "TOPADMIN" inputs NetEx/eFT script "VAX.UA" during program execution, and script file "VAX.UA" contains the following line:

```
CONNECT vax vaxuser secret
```

To avoid storing the password “SECRET” in cleartext form in the script file, the password is encrypted using the following alias command from an EFT command prompt:

```
encrypt secret topadmin
```

Username “TOPADMIN” is specified because that is the local MVS username under which the NetEx/eFT that uses the connect/login information is run.

ENCRYPT returns an encrypted form of the password in the global variable pw:

```
*288d6d90f303fe84c
```

Using a text editor, the user enters the encrypted form of the password in the file “VAX.UA”, resulting in:

```
CONNECT vax vaxuser *288d6d90f303fe84c
CONNECT vax vaxuser {pw}
```

# Appendix B. NetEx/eFT Error Messages for IBM z/OS

This appendix is intended to give users more information about NetEx/eFT messages that may be seen during a session. Many of the messages are self-explanatory (e.g., "Invalid command") and require no further discussion. At the end of this message table is a list and description of the messages that require further explanation.

It should be noted that the NetEx/eFT messaging scheme is designed to generate various levels of messages which is why a single erroneous condition may result in two or more messages. Each level of message that is displayed (from first to last), is designed to be slightly more specific than the message preceding it. All of the messages that are displayed should be considered when attempting to diagnose an error condition. Each message is prefixed with a Facility or system name that generated the message making the messages unique:

- UA- NetEx/eFT host independent message
- EFT213- NetEx/eFT eFT213 IBM z/OS host dependent message
- MUX- NetEx/eFT Multiplex Server
- MUX213- NetEx/eFT eFT213 IBM z/OS MUX Server
- UAxx3, MUXxx3, NETEX etc. - look in the appropriate NetEx or eFT manual for that host (e.g., Hxx3 NetEx/eFT, NetEx/IP for zOS H210IP).

Also, in the section following the table are additional description of some of the messages.

Below is a breakdown of each column of the message table along with a description of the entries that may appear under it.

**Code** The unique error or message code.

**Sev** A single character severity level indicator. The possible values are I (Information), W (Warning), E (Error), or F (Fatal).

**Comment** A list of zero to five characters giving more details about the message. If no comment characters are given, the message (along with accompanying messages), is intended to be self-explanatory. The possible Comment characters are:

**A** An additional description of the message is given at the end of the message table.

**D** Diagnostic or Internal error. These errors are very unlikely to occur and may indicate a more severe problem is at hand or that some unexpected internal condition occurred. The user should refer to accompanying messages if displayed, for a further explanation of the problem. These messages should be logged and reported to the system administrator.

**H** Host specific messages will accompany these messages. The host specific messages should provide additional information as to the cause of the condition.

**I** Internal error; contact NESi Support.

**N** Network related message. In a NETEX environment, the user should refer to the accompanying NETEX message that is displayed for a further explanation of the problem. These messages should be logged and reported to the system administrator unless it is obvious that the NETEX condition is temporary and for a known reason. These messages could possibly be a sign of a network interruption of some kind.

**R** Retriable error condition. The command used to generate this error can be re-tried at some later time without fear of a fatal condition occurring.

**Text** The message text. In the following message texts, SSS indicates a string value is present in the actual message; NNN, a decimal value; XXX, a hexadecimal value and S, a single character. **Most Error messages are self-explanatory, however if more information has been provided look in the section following this table.**

**Table 5. Error Messages**

| Code | Severity | Comment | Text |
|---|---|---|---|
| 201 | E | | "Invalid positional parameter '%s'" |
| 202 | E | | "Invalid command line switch '%s'" |
| 203 | E | | "Missing value for switch '%s'" |
| 204 | E | | "Invalid numeric '%s' for switch '%s'" |
| 301 | E | I | "ENV_PUT: missing keyword" |
| 302 | E | A | "Overflow of %d byte environment buffer" |
| 303 | E | A | "Failed to add '%s=%.15s...'" |
| 304 | E | | "Environment concatenate failure" |
| 501 | E | A D N | "Protocol error - expected '%c%c' - got '%c%c'" |
| 518 | I | | "RNT - Restarting failed file transfer, timeout = %s seconds." |
| 519 | I | | "RNT - Retransmitting %ld bytes." |
| 520 | I | | "RNT - Restart complete." |
| 601 | E | I | "Attempt to free active NRB" |
| 602 | E | | "Netex blocksize negotiation failed (%d)" |
| 605 | E | R | "CONFIRM timed out after %d seconds" |
| 606 | E | R | "Error on CONFIRM from service '%s'" |
| 607 | E | R | "Missing CONFIRM indicator (%d)" |
| 608 | E | | "Offer of service '%s' timed out after %d seconds" |
| 609 | E | R | "Failed to OFFER service '%s'" |
| 610 | E | R | "Missing CONNECT indicator (%d)" |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 611 | E | R | “CONFIRM of offer failed” |
| 612 | E | R I | “NET_READ: ATTENTION read failed” |
| 613 | E | R | “Missing NORMAL data indicator (%d)” |
| 614 | E | R | “Buffer length %d less than ATTENTION message %d” |
| 701 | E | | “Protocol buffer size %d is less than minimum %d” |
| 702 | E | I | “PROT_OPEN: env send” |
| 703 | E | I | “PROT_OPEN: env recv” |
| 704 | E | D | “Failed to get protocol keyword value for %s” |
| 705 | E | R I | “PROT_CONTROL: non-wait read” |
| 706 | E | | “Protocol record (%d) is larger than buffer size (%d)” |
| 707 | E | I | “PROT_SEND: attn flush” |
| 708 | E | I | “PROT_FLUSH: flush %d bytes” |
| 709 | E | I | “PROT_WRITE: %d unflushed records” |
| 710 | E | R I | “PROT_WRITE: attn read” |
| 711 | E | R I | “PROT_WRITE: write %d bytes” |
| 712 | E | D | “Failed to read STDIN” |
| 713 | E | D N | “Invalid protocol Record flag ‘%c’” |
| 714 | E | D | “Protocol buffer (%d) is too small for record (%d)” |
| 715 | E | I | “PROT_FILL: %d byte read” |
| 716 | E | I | “PROT_FILL: bad mode (%d bytes)” |
| 717 | E | D N | “Invalid protocol Block/Record flag ‘%c’” |
| 718 | E | I | “PROT_READ: flow XON” |
| 720 | E | I | “PROT_READ: attn ack” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 721 | E | R I | “PROT_READ: read %d bytes” |
| 722 | E | I | “PROT_MSG_CHECK: %s” |
| 723 | E | | “Failed to receive INFORMATIVE messages” |
| 724 | E | I | “PROT_MSG_CHECK - %s” |
| 725 | E | | “Buffer (%d) too small for ATTENTION message (%d)” |
| 730 | E | I | “PROT2_FILL: %d byte header” |
| 731 | E | | “Invalid protocol Block flag ‘%c’” |
| 732 | E | | “Block size %d is greater than buffer size %ld” |
| 733 | E | I | “PROT2_FILL: %d byte read” |
| 801 | E | D | “Missing HOST name” |
| 802 | E | D | “Missing SERVICE name” |
| 803 | E | R | “Service ‘%s’ is not offered on host ‘%s’” |
| 804 | E | | “Host ‘%s’ does not exist in configuration” |
| 805 | E | N R | “Error connecting to service ‘%s’ on host ‘%s’” |
| 808 | E | N R | “Error on WRITE to service ‘%s’” |
| 809 | E | N R | “READ timed out after %d seconds” |
| 810 | E | N R | “Error on READ from service ‘%s’” |
| 811 | E | D N R | “Bad data length %d on READ” |
| 812 | E | D R | “Bad DATAMODE ‘%d’ on exchange” |
| 813 | E | D N | “Failed to open protocol connection” |
| 814 | E | D N | “Failed to send CONNECT environment” |
| 815 | E | D N | “Failed to receive CONNECT response” |
| 816 | E | D N | “Failed to close protocol connection” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 817 | E | D | “Missing SERVICE name” |
| 821 | E | N R | “READ timed out after %d seconds” |
| 822 | E | N R | “Error on READ of datamode” |
| 823 | E | D N R | “Bad data length %d on READ” |
| 824 | E | D R | “Bad DATAMODE ‘%d’ on exchange” |
| 825 | E | N R | “Error on WRITE of datamode” |
| 826 | E | D N | “Failed to open protocol connection” |
| 827 | E | D N | “Failed to receive CONNECT environment” |
| 828 | E | R | “Invalid process identifier (PID) on reconnect” |
| 829 | E | D N | “Failed to send CONNECT response” |
| 830 | E | D N | “Failed to close protocol connection” |
| 831 | E | | “Failed to decrypt the password” |
| 832 | E | | “Remote server does not support gateway (SI+) protocol” |
| 901 | E | | “Invalid encrypted password” |
| 902 | E | | “Bad HEX string” |
| 903 | E | | “Encryption key is a null string” |
| 904 | E | | “Missing secondary key for password decryption” |
| 905 | E | | “Invalid primary encryption key” |
| 906 | E | | “Invalid secondary encryption key” |
| 907 | E | | “Missing alternate primary key for password decryption” |
| 2001 | E | I | “CRC_READ: %08lx vs. %08lx” |
| 2002 | E | A R | “%s error at block %ld” |
| 2003 | E | I | “CRC_READ: %ld vs. %ld” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 2004 | E | A R | “Sequence number error at block %ld” |
| 2101 | E | A | “Failed to allocate %ld bytes of dynamic memory.” |
| 2401 | E | | “Missing subtask work area address” |
| 2402 | E | | “Invalid subtask work area address” |
| 2501 | E | | “Invalid TCP host name ‘%s’” |
| 2502 | E | A | “Invalid TCP service name ‘%s’” |
| 2503 | E | | “Failed to connect to TCP service ‘%s’ on host ‘%s’” |
| 2505 | E | | “Failed to bind to TCP service ‘%s’” |
| 2506 | E | | “Failed to listen for TCP c” |
| 2507 | E | | “Offer of service ‘%s’ timed out after %d seconds” |
| 2508 | E | | “Failed to accept TCP connection” |
| 2509 | E | I | “NET_READ: tag=%d length=%d mode=%d buf=%.32s” OR “NET_READ: tag=%d length=%d mode=%d” |
| 2510 | E | | “Bad 32-byte block mode header detected” |
| 2511 | E | | “Buffer (%d) too small for pending data (%d)” |
| 2512 | E | | “Select of %d TCP sockets failed (max_fd=%d)” |
| 2513 | E | | “Failed to create a TCP socket” |
| 2515 | E | | “Unsupported remote datamode (%d)” |
| 2516 | E | | “TCP socket close detected - read length of zero” |
| 2517 | E | | “Failed to set REUSEADDR socket option” |
| 2518 | E | | “Failed to set TCP %s socket option to %d” |
| 2519 | E | | “The TCP connection was closed prematurely” |
| 2520 | E | | “Failed to read %d bytes of TCP %s” |
| 2521 | E | | “Failed to write %d bytes of TCP %s” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 2522 | E | | “SELECT failed for TCP socket %d - timer %d” |
| 2523 | E | | “Failed to get assigned TCP port number” |
| 2535 | E | | “Unable to assign a TCP port in the range %d-%d” |
| 2536 | E | N | “Failed to bind dynamic TCP port ‘%d’ - %d” |
| 2551 | E | | “Invalid TCP host name ‘%s’” |
| 2552 | E | H | “Failed to get local host name” |
| 2554 | E | | “Invalid port returned: ‘0’” |
| 3000 | E | | “NET_CONNECT(%d): setup_mvsgsk_env error, errno=%d” |
| 3100 | I | | “Support documentation messages” |
| 3101 | E | | “MVS:Fipslvl can on be set once, and turned off. fipslvl=n” |
| 3102 | E | A | “The value for certlb was not found in the GSK database” |
| 3103 | E | A | “The value for certpw did not match the password for the database” |
| 3104 | E | A | “The dataset specified by certdb was not found” |
| 3105 | E | A | “Connection was reset by peer. Check for -secure -nosecure settings” |
| 3106 | E | A | “Socket init failed Check for -secure -nosecure settings” |
| 4001 | E | I | “ALIAS_CREATE: missing alias” (alias creation) |
| 4001 | W | A | “Invalid CONTROL password” (server) |
| 4001 | W | | “Invalid OPERATOR password” (svcinit) |
| 4002 | E | I | “ALIAS_COMMAND: missing command” (alias) |
| 4002 | W | A | “Multiplexed Server stopped” (server) |
| 4002 | W | | “Service Initiator stopped” (svcinit) |
| 4003 | W | | “Alias ‘%s’ is not defined” (alias) |
| 4003 | W | A | “Trace flag settings: %s” (server) |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 4004 | W | A | “INFO not implemented” |
| 4005 | W | A | “Invalid CONTROL request ‘%s’” |
| 4011 | E | A R | “Sorry - %s session limit exceeded (%d active)” |
| 4012 | E | A | “Failed to login as user ‘%s’” |
| 4013 | E | I | “MSERVER: send sync” |
| 4014 | E | A | “A second network connection is required” |
| 4015 | E | A | “Incompatible EFT version for file transfer” |
| 4016 | E | A | “Invalid code table length %d” |
| 4017 | E | A | “Unsupported request code ‘%c’ - no action taken” |
| 4018 | F | A | “Failed to invoke helper %s” |
| 4019 | F | A | “Timeout during helper processing” |
| 4021 | E | A | “Server RESULTS failed (type=%c status=%d)” |
| 4100 | E | | “Connecting to a Terminal Emulation Server requires ” “the -FULL qualifier” |
| 4101 | E | I | “CLIENT: restart flush” |
| 4101 | E | I | “SCRIPT_OUTPUT: send login info” |
| 4101 | W | | “Invalid CONTROL password” (server) |
| 4102 | E | | “BLOCKSIZE of %d is out of range (%d-%d)” (client) |
| 4102 | W | | “Terminal Emulation Server stopped” (server) |
| 4103 | W | | “Cannot have more than %d active connections” (client) |
| 4103 | W | | “Trace flag settings: %s” (server) |
| 4104 | E | D | “Missing connect SERVICE” (client) |
| 4104 | W | | “INFO not implemented” (server) |
| 4105 | E | | “Failed to connect service ‘%s’ on host ‘%s’” (client) |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 4105 | W | | “Invalid CONTROL request ‘%s’” (server) |
| 4106 | E | | “TSERVER: connect quit” (server) |
| 4106 | I | A | “The requested network blocksize %d was reduced to %d” (client) |
| 4107 | E | D | “Failed to open CLIENT protocol connection” (client) |
| 4107 | E | I | “TSERVER: send ECB” (server) |
| 4108 | E | D N | “Failed to receive connect information” (client) |
| 4108 | E | I | “TSERVER: send ECB defaults” (server) |
| 4109 | E | I | “TSERVER: send ECB keyword” (server) |
| 4109 | I | A | “There were %d CONNECT records ignored” (client) |
| 4110 | E | N | “Failed to DISCONNECT” (client) |
| 4110 | E | | “Failed to establish the Terminal Emulation connection” (server) |
| 4111 | E | | “Sorry - %s session limit exceeded (%d active)” |
| 4112 | W | | “Remote host required for remote help request” |
| 4113 | E | N | “Failed to request remote RECEIVE” (client) |
| 4113 | E | I | “TSERVER: send sync” (server) |
| 4114 | E | N | “Failed to get SEND acknowledge” |
| 4116 | E | N | “Failed to request remote SEND” |
| 4117 | E | N | “Failed to get RECEIVE acknowledge” (client) |
| 4117 | E | | “Unsupported request code ‘%c’ - no action taken” (server) |
| 4118 | E | N | “Failed to send SOURCE/DEST environments” |
| 4121 | E | | “Server RESULTS failed (type=%c status=%d)” |
| 4121 | W | | “Missing remote command” (client) |
| 4122 | E | | “Request is not supported in terminal emulation mode” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 4123 | E | | “Data block format error” (server) |
| 4123 | E | N | “Failed to receive an ABORT acknowledge” (client) |
| 4124 | E | | “Remote %s failed” (client) |
| 4124 | E | | “Unexpected network data packet type of %X” (server) |
| 4125 | E | | “Required encryption is not supported” (server) |
| 4125 | E | I | “CLIENT: results %s failed” (client) |
| 4126 | W | D | “Command ‘%s’ is not implemented” |
| 4127 | E | A D | “The MESSAGE stack is empty” |
| 4128 | E | N | “Failed to send error message to remote server” |
| 4129 | E | N | “Failed to communicate with remote server” |
| 4130 | E | I | “CLIENT_RESYNC: %s failed” |
| 4131 | E | A N | “Failed to establish secondary network connection” |
| 4132 | E | A | “Restricted command in server startup file” |
| 4135 | E | A,D | “License Verify Error” |
| 4136 | W | A,D | “License Expiring Warning” |
| 4137 | E | A,D | “Product License protocol CAPability is incorrect.” |
| 4138 | W | A,D | “License Verify Capability Warning” |
| 4139 | W | A | “License expired SSSS, product will cease to function SSSS” |
| 4140 | E | A | “License expired, product disabled” |
| 4141 | E | | “Secure CONNECT is not licensed” |
| 4142 | F E | | “Secure SERVER is not licensed” |
| 4201 | E | | “Missing %s parameter” |
| 4202 | E | | “Invalid %s parameter ‘%s’” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 4203 | W | | “There is no active remote host” |
| 4301 | E | | “Failed to open SIHELPER protocol connection” |
| 4301 | E | | “Help is not available for ‘%s’” |
| 4302 | E | | “Failed to send SIHELPER connect environment” |
| 4302 | E | | “Help line is longer than %d characters” |
| 4303 | E | | “Failed to receive SIHELPER connect response” |
| 4304 | E | | “Login request to SIHELPER failed” |
| 4501 | E | A | “Nested (or recursive) input/alias limit of %d exceeded” |
| 4502 | F | D | “Can’t open STDOUT” |
| 4503 | E | | “Output PREFIX (%d) exceeds COLUMNS (%d)” |
| 4504 | E | A | “Bad output FORMAT definition - reset to default” |
| 4505 | E | | “Input request (%d byte maximum) failed” |
| 4601 | W | | “Variable ‘%s’ contains invalid characters” |
| 4602 | W | | “Variable ‘%s’ is longer than %d characters” |
| 4603 | W | | “Qualifier %s cannot be modified” |
| 4604 | W | | “A value is required for qualifier %s” |
| 4605 | W | | “Invalid %s numeric value ‘%s’” |
| 4606 | W | | “%s value %s is out of range (%s)” |
| 4607 | W | | “Invalid %s boolean value ‘%s’” |
| 4608 | E | I | “KEY_CHECK: request” |
| 4609 | E | I | “KEY_CHECK: results” |
| 4610 | W | | “Invalid %s option ‘%s’” |
| 4701 | W | A | “Recursive alias ‘%s’” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 4702 | W | | “There is no active remote host” |
| 4703 | W | | “Invalid %s qualifier ‘%s’” |
| 4704 | W | A | “Use SET LOCAL/REMOTE to modify %s qualifier ‘%s’” |
| 4705 | W | | “Missing value for %s qualifier ‘%s’” |
| 4706 | W | | “Too many parameters for %s” |
| 4707 | W | | “%s requires additional parameters” |
| 4708 | W | | “Invalid command ‘%s’” |
| 4709 | W | A | “Command token is greater than %d characters” |
| 4802 | E | D | “Missing MAXRECORD specification” |
| 4803 | E | D | “MAXRECORD (%ld) greater than maximum (%ld)” |
| 4804 | E | A | “MAXRECORD (%ld+%ld) too large for BLOCKSIZE (%d)” |
| 4807 | E | D | “Invalid datamode (%d) in record RECEIVE” |
| 4808 | E | D N | “Bad header flag (%d) in record RECEIVE” |
| 4809 | E | A D N | “Sequence error (%d vs. %d) in record RECEIVE” |
| 4812 | E | D | “Missing MIN_BYTE_COUNT specifier” |
| 4901 | E | | “Failure during %s mode receive” |
| 4902 | E | I | “RECV_FILE: %s” |
| 4903 | E | | “Failure during RECEIVE file setup” |
| 4904 | E | I | “RECV_WILD: %s” |
| 4906 | E | | “Receiving process exited with data still pending” |
| 5001 | E | | “Failure during %s mode send” |
| 5002 | E | I | “SEND_FILE: %s” |
| 5003 | E | | “Failure during SEND file setup” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 5004 | E | I | “SEND_WILD: %s” |
| 5006 | E | | “Wild card names cannot be used with segmenting options” |
| 5007 | E | | “No pound sign (#) found in segmented file template” |
| 5008 | E | | “SEG_SIZE cannot be used with SEG_START and SEG_END” |
| 5010 | E | | “Not all FILELIST files found. Could not find %s” |
| 5101 | E | I | “SERVER: ecb” |
| 5102 | E | I | “SERVER: ecb defaults” |
| 5103 | E | I | “SERVER: ecb keyword” |
| 5104 | E | I | “SERVER: ecb alias” |
| 5105 | E | I | “SERVER: connect startup” |
| 5106 | E | I | “SERVER: connect quit” |
| 5107 | F | N | “Failed to receive next CLIENT request” |
| 5108 | E | I | “SERVER: send sync” |
| 5109 | E | | “Server %s failed” |
| 5110 | E | I | “SERVER: results %s failed” |
| 5111 | E | | “Keyboard interrupt” |
| 5112 | E | | “Keyboard interrupt” |
| 5113 | E | I | “SERVER: source” |
| 5114 | E | I | “SERVER: dest” |
| 5115 | E | D | “Missing VALIDATE qualifier” |
| 5116 | E | D | “Invalid VALIDATE qualifier ‘%s’” |
| 5117 | E | I | “SERVER: validation results” |
| 5118 | E | D N | “Invalid code table length %d” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 5119 | E | D | “Unsupported request code ‘%c’ - no action taken” |
| 5120 | E | A D | “The MESSAGE stack is empty” |
| 5121 | F | N | “Failed to send error message to CLIENT” |
| 5122 | F | N | “Failed to offer service ‘%s’” |
| 5123 | F | D | “Failed to open SERVER protocol connection” |
| 5124 | F | | “Missing or invalid password” |
| 5125 | E | | “Failed to decrypt -PASSWORD” |
| 5125 | E | | “Remote command execution is not allowed” |
| 5201 | W | | “Missing HOST specifier” |
| 5202 | W | | “Invalid HOST index ‘%d’” |
| 5203 | W | | “Host ‘%s’ is not active” |
| 5204 | W | | “Missing %s qualifier” |
| 5205 | W | | “Invalid %s qualifier ‘%s’” |
| 5206 | W | A | “Use SET LOCAL/REMOTE to modify %s qualifier ‘%s’” |
| 5207 | W | | “Invalid %s qualifier ‘%s’” |
| 5208 | W | | “There are no remote host connections” |
| 5301 | E | A | “Invalid allocation parameters (%ld + %ld > %ld)” |
| 5303 | I | | “Character code translation was NOT performed” |
| 5304 | E | A | “Invalid ARCHIVE file format ‘%.4s’” |
| 5305 | E | A | “Invalid ARCHIVE block length (%ld)” |
| 5306 | I | A | “Incomplete ARCHIVE file - missing end-of-file” |
| 5307 | E | | “Incomplete ARCHIVE write (%d of %d bytes written)” |
| 5308 | E | | “Record buffer overflow (%d) during TAB expansion” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 5310 | E | | “ARCHIVE file cannot be filtered” |
| 5311 | E | | “Cannot restart resilient transfer, increase RNT_BUFalloc to %ld” |
| 5312 | E | | “Cannot restart resilient transfer, no memory” |
| 5313 | E | | “ARCHIVE filtering error (segment %d / record %ld / byte %.0f)” |
| 5314 | E | | “ARCHIVE filtering error (segment %d / record %ld / byte %.0f)” |
| 5315 | E | | “Searching for sequence number %ld” |
| 5401 | W | A | “More than %d levels of nested strings” |
| 5402 | W | | “Unmatched string delimiter ‘%c’” |
| 5403 | W | | “Invalid string format” |
| 5404 | W | | “Unmatched string delimeter ‘%c’” |
| 5405 | W | | “No closing quote on string literal” |
| 5406 | W | A | “Empty string substitution” |
| 5407 | W | | “String variable ‘%s:%s’ is invalid” |
| 5410 | W | | “Invalid string function ‘%s’” |
| 5411 | W | | “Invalid parameter for function ‘%s’” |
| 5412 | W | | “Too many parameters for function ‘%s’” |
| 5413 | W | | “Invalid numeric parameter ‘%s’ for function ‘%s’” |
| 5414 | W | | “Missing parameters for function ‘%s’” |
| 5415 | W | D | “String function ‘%s’ is not supported” |
| 5416 | E | | “Failed to encrypt password” |
| 5501 | E | N | “Failed to send DEBUG information” |
| 5601 | E | D | “TRANSLATE trouble - invalid sequence” |
| 5602 | E | | “TRANSLATE (code conversion) not supported by remote host” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 5603 | W | A | “Character code cannot be translated” |
| 5604 | E | D N | “Failed to capture Netex code conversion table” |
| 5605 | E | D N | “Invalid translate code table length %d” |
| 5606 | E | N | “Failed on binary read of Netex code table (%d)” |
| 5607 | E | N | “Failed to exchange TRANSLATE information” |
| 5608 | W | | “Invalid TRANSLATE value ‘%s’” |
| 5609 | W | | “TRANSLATE value %s is out of range (0-%d)” |
| 5610 | W | | “Translate IN_ONLY and OUT_ONLY are both ‘on’” |
| 5701 | E | | “Duplicate label ‘%s’” |
| 5702 | E | | “Label ‘%s’ is not 1-%d characters in length” |
| 5703 | E | | “Label ‘%s’ contains invalid characters” |
| 5704 | E | | “Missing label ‘%s’” |
| 6301 | E | | “Invalid transfer mode for compress/expand option” |
| 6302 | E | | “Unsupported compression method (%d)” |
| 6303 | I | | “Ignoring EXPAND qualifier - the data is not compressed” |
| 6304 | E | | “Incompatible release for CHARACTER mode compress/expand” |
| 6305 | E | | “Cannot code convert compressed data (%d)” |
| 6401 | E | | “LZW bits (%d) outside the range 12-%d” |
| 6403 | I | | “Not compressing due to insufficient memory” |
| 6501 | E | | “Cannot obtain current USERNAME” |
| 6502 | E | | “Failed to encrypt password” |
| 6503 | E | H | “Failed to create output file ‘%s’” |
| 6504 | E | | “TSO/E output failed, code (%d)” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 6505 | E | | “Invalid character input” |
| 6506 | E | | “Character input longer than (%d) bytes” |
| 6507 | E | | “Character input failed for (%d) bytes” |
| 8000 | E | | “Remote login not implemented for IBM/MVS” |
| 8001 | E | A | “Missing APPL_ID” (server) |
| 8001 | E | | “Missing USERNAME” |
| 8001 | E | A | “Keyboard interrupt” |
| 8002 | E | A R | “All connections to TSO are busy” |
| 8002 | E | | “Subtask attach failed” |
| 8002 | E | A | “File size limit exceeded” |
| 8003 | E | A | “Could not find PROCESS: %s” |
| 8003 | E | | “Login failed - unable to attach Server subtask” |
| 8004 | E | | “LOGIN_NODE not implemented for IBM/MVS” |
| 8004 | E | A | “Login exceeded %d second timeout” |
| 8005 | E | A | “Missing USERNAME” |
| 8005 | E | | “Server subtask terminated early, code 0x%08.8X” |
| 8006 | E | | “Login message buffer size (%d) exceeded” |
| 8006 | E | A | “No response supplied to a LOGIN prompt” |
| 8007 | E | | “Login message buffer incomplete” |
| 8007 | E | I | “PATH_LOGIN: login results” |
| 8008 | E | A | “Failed to connect to TSO” |
| 8008 | E | | “Login exceeded %d second timeout” |
| 8009 | E | | “Read from Server subtask failed” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8009 | E | I | “PATH_LOGIN: Invalid state (%d)” |
| 8010 | E | | “Missing Server command” |
| 8011 | E | | “Detach of subtask failed, code %d” |
| 8011 | E | A | “Failed to execute command: %.30s” |
| 8012 | E | | “Unable to start login timer” |
| 8012 | F | A | “VTAM connection lost” |
| 8013 | E | | “Maximum subtask limit exceeded” |
| 8013 | E | I | “PATH_PUT: Out of buffers” |
| 8014 | E | I H | “PATH_PUT: VTAM buffer get failed” |
| 8020 | E | | “Invalid SYSOUT class ‘%c’” |
| 8021 | E | | “Unable to allocate Server SYSOUT data set” |
| 8022 | E | | “Unable to deallocate Server SYSOUT data set” |
| 8023 | E | | “SVC99_INITIALIZE failed” |
| 8024 | E | | “SVC99_TEXT_UNIT failed, code %d” |
| 8025 | E | | “SVC99_FREE failed” |
| 8101 | E | | “Missing APPL_ID” |
| 8102 | E | | “All VTAM connections are busy” |
| 8103 | E | | “Could not find application %s” |
| 8104 | E | | “Login exceeded %d second timeout” |
| 8105 | E | | “Failed to connect” |
| 8106 | E | I | “TPATH_LOGIN: Invalid state (%d)” |
| 8107 | F | | “VTAM connection lost” |
| 8108 | E | I | “TPATH_PUT: Out of buffers” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8109 | E | I H | “TPATH_PUT: VTAM buffer get failed” |
| 8201 | E | A | “%s command execution is not supported” |
| 8202 | E | A R | “Command input error” |
| 8203 | I | A | “Note: %s TSOPREFIX may not match DIRECTORY” |
| 8300 | E | A | “Failed to access data set %s” |
| 8301 | E | A | “Failed to create data set %s%s%s” |
| 8302 | E | A | “Failed to delete data set %s” |
| 8303 | E | A | “No data set name or DD name was specified” |
| 8303 | E | I | “FILE_GDG: No data set name or DD name was specified” |
| 8304 | E | A | “Data set not found in catalog” |
| 8304 | I | I | “Data set not found in catalog” |
| 8305 | E | A | “Cataloged data set not found on volume %s” |
| 8306 | E | A | “Data set not found on requested volume %s” |
| 8306 | E | | “Data set not found on requested volume” |
| 8307 | E | A | “DD name %s is not allocated” |
| 8308 | E | A | “Cannot process concatenated DD name %s” |
| 8309 | E | A | “Volume %s is not available” |
| 8310 | E | I | “FILE_ACCESS: Error initializing ” “dynamic allocation control blocks” |
| 8310 | E | I | “FILE_CLOSE: Error initializing ” “dynamic al-location control blocks” |
| 8310 | E | I | “FILE_CREATE: Error initializing ” “dynamic allocation control blocks” |
| 8310 | E | I | “FILE_DELETE_MVS: Error initializing ” “dynamic allocation con-trol blocks” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8312 | E | A | “Unrecoverable input/output error, ” “check DCB parameters” |
| 8313 | E | A | “Unrecoverable input/output error ” “reading partitioned data set directory” |
| 8314 | E | A | “MVS ABEND code S%03X-%02X” |
| 8314 | E | I | “FILE_ABEND: IBM/MVS ABEND code S%03X-%02X” |
| 8315 | E | I | “FILE_FLUSH: Failed for (%d) bytes” |
| 8316 | E | I | “FILE_ACCESS: MVS_DATASET_OPEN error, code(%d, %d, %d)” or “FILE_CREATE: MVS_DATASET_OPEN error, code(%d, %d, %d)” |
| 8317 | E | | “Delete of member %s failed” or “Delete of member %s failed, code (%d)” or “Delete of member %s failed, code (0x%08.8X)” |
| 8318 | E | I | “FILE_ACCESS: MVS_DATASET_CLOSE error, code(%d, %d, %d)” |
| 8319 | E | I | “FILE_ACCESS: MVS_DATASET_INFO error, code(%d, %d, %d)” |
| 8320 | E | I | “FILE_GET: MVS_DATASET_GET error, code(%d, %d, %d)” or “FILE_REC_READ: MVS_DATASET_GET error, code(%d, %d, %d)” |
| 8321 | E | I | “FILE_PUT: MVS_DATASET_PUT error, code(%d, %d, %d)” or “FILE_REC_WRITE: MVS_DATASET_PUT error, code(%d, %d, %d)” |
| 8322 | E | I | “FILE_ACCESS: MVS_PDS_MEMBER_FIND error, ” “code(%d, %d, %d)” or “FILE_CREATE: MVS_PDS_MEMBER_FIND error, ” “code(%d, %d, %d)” |
| 8323 | E | I | “FILE_GET: Failed for (%d) bytes” |
| 8324 | E | A | “Data set exists and -CREATE NEW was specified” |
| 8325 | E | A | “Member %s exists and -CREATE NEW was specified” |
| 8326 | E | A | “Member %s does not exist ” “in partitioned data set directory” or “Member does not exist ” “in partitioned data set directory” |
| 8327 | E | A | “Text record longer than %d bytes” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8328 | E | A | “Data set contains non-character data” |
| 8329 | E | A | “Character input longer than %d bytes” |
| 8330 | E | A | “DESTINATION name contains too many wildcard characters” |
| 8331 | E | A | “A valid SPACE value is required” “to create this data set” |
| 8332 | E | A | “-CREATE APPEND is not valid for a ” “partitioned data set” |
| 8333 | E | A | “-CREATE APPEND is not valid for a DD name” |
| 8334 | E | A | “-CREATE DELETE is not valid for a ” “partitioned data set” |
| 8335 | E | A | “-CREATE DELETE is not valid for a DD name” |
| 8336 | E | A | “Password required” |
| 8337 | E | A | “Record longer than maximum allowed of %d” |
| 8338 | E | A | “Error encountered while reading from data set” |
| 8339 | E | A | “Error encountered while writing to data set” |
| 8340 | E | I | “FILE_PUT: Failed for (%d) bytes” |
| 8341 | E | I | “FILE_FLUSH: TSO output failed, code (%d)” |
| 8341 | E | I | “FILE_INPUT: TSO output failed, code (%d)” |
| 8341 | E | I | “FILE_PUT: TSO output failed, code (%d)” |
| 8342 | E | A | “RECFORMAT (%s) is not supported for CHARACTER ” “mode operations” |
| 8343 | E | A | “RECFORMAT (%s) is not supported for RECORD ” “mode operations” |
| 8344 | E | A | “RECFORMAT (%s) is not supported for STREAM ” “mode operations” |
| 8345 | E | A | “Data set does not have a valid ” “RECFORMAT” |
| 8346 | E | A | “RECFORMAT (%s) is not supported” |
| 8347 | E | A | “Record length (%d) exceeds maximum record length (%d)” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8348 | E | A | “RECLENGTH (%d) is out of range (%d - %d)” |
| 8349 | E | A | “BLOCKSIZE (%d) is out of range (%d - %d)” |
| 8350 | E | A | “BLOCKSIZE (%d) must be a multiple ” “of RECLENGTH (%d) for RECFORMAT (%s)” |
| 8350 | E | | “Failed to add ‘%s=%.40s...’” |
| 8351 | E | A | “BLOCKSIZE (%d) must equal RECLENGTH (%d) ” “for RECFORMAT (%s)” |
| 8351 | E | | “Overflow of %d byte environment buffer” |
| 8352 | E | | “Data set does not have a valid organization” |
| 8353 | E | A | “Data set organization (%s) is not supported” |
| 8354 | E | A | “Spanned records are not supported for ” “partitioned data sets” |
| 8355 | E | A | “Standard records are not supported for ” “partitioned data sets” |
| 8356 | E | | “Missing SOURCE file specification” |
| 8357 | E | A | “Member name invalid for a sequential data set” |
| 8358 | E | A | “Member name omitted for a partitioned data set” |
| 8359 | E | A | “Zero length record invalid for a data set ” “with printer control” |
| 8360 | E | A | “Primary space filled and ” “no secondary space available” |
| 8361 | E | A | “Space on the current volume filled and ” “no more volumes available” |
| 8362 | E | A | “Space on the current volume filled and ” “next volume unavailable” |
| 8363 | E | A | “Space on the current volume filled and ” “next volume has too many users” |
| 8364 | E | A | “Space on the current volume filled and ” “installation exit rejected next volume” |
| 8365 | E | A | “%s is longer than maximum of %d character(s)” or “Data set name is longer than ” “maximum of 44 character(s)” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8366 | E | A | “Invalid or unexpected character ‘%c’ found in %s” |
| 8367 | E | A | “Requested MODE is not supported” |
| 8368 | E | A | “Unmatched apostrophes in data set name” |
| 8369 | E | A | “Unmatched parenthesis in data set name” |
| 8370 | E | A | “Bad qualifier in data set name” |
| 8371 | E | A | “Partitioned data set directory is full” |
| 8372 | E | I | “FILE_ARCH_READ: bad tag character ‘%c’” |
| 8373 | E | A | “Unknown archive mode tag field ‘%.4s’” |
| 8374 | E | A | “Invalid ARCHIVE tag field ‘%.4s’” |
| 8375 | E | A | “Bad ARCHIVE record length %d” |
| 8376 | I | A | “Note: %d record(s) truncated at %d characters” |
| 8377 | I | A | “Note: %d record(s) wrapped at %d characters” |
| 8378 | I | A | “Note: %d record(s) padded to %d characters” |
| 8379 | E | A | “Tape label data set name mismatch” |
| 8380 | E | A | “Invalid password” |
| 8381 | E | A | “You are not authorized to access this data set” |
| 8382 | I | | “Note: ISPF statistics were not created” |
| 8383 | E | | “-RETAIN is not valid unless -DDNAME is provided” |
| 8384 | E | | “-REFER * is not valid unless -DDNAME is provided” |
| 8385 | E | | “-DDNAME is not valid with a DD name file specification” |
| 8386 | E | | “The data set has not reached its expiration date” |
| 8387 | E | | “The request to label the volume was rejected” |
| 8388 | E | | “Unrecoverable input/output error ” “writing partitioned data set directory” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8389 | E | | “The data set has invalid attributes, ” “check DCB parameters” |
| 8390 | E | | “ISO/ANSI/FIPS labels not supported” |
| 8391 | E | | “Unable to open the SYSIN/SYSOUT data set” |
| 8392 | E | | “The requested subsystem is not available” |
| 8393 | E | | “GDG base entry not found in catalog” |
| 8394 | E | | “Missing or invalid relative GDG number” |
| 8395 | E | | “Unable to process requested GDG” |
| 8396 | I | I | “MVS LOCATE failed with return code %d, ” “refer to MVS message IDC3009I” |
| 8398 | E | | “Catalog did not contain any volume information” |
| 8399 | E | | “EXPIRATION and RETENTION qualifiers ” “cannot be specified together” |
| 8400 | E | | “Invalid EXPIRATION date (use YYDDD or YYYYDDD).” |
| 8401 | E | | “LABELNUMBER not equal to one and ” “no volume serial specified” |
| 8402 | E | | “LABELNUMBER not equal to 1” or “LABELNUMBER not equal to 1 or %d” |
| 8403 | E | I | “FILE_CREATE: LABELNUMBER not 1 and ” “no retained data set name found” |
| 8404 | I | I | “Volume information obtained from data set ‘%s’” |
| 8405 | I | I | “Volume information not retained since ” “DISPOSITION is not CATALOG” |
| 8406 | E | | “Zero length record invalid for a data set ” “with undefined records” |
| 8407 | E | | “Data set resides on a migration or archival volume” |
| 8407 | I | I | “Data set resides on a migration or ” “archival volume” |
| 8410 | E | | “Missing DESTINATION file specification” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8420 | E | | “Invalid file sequence number for this volume” |
| 8421 | E | | “Invalid record length for this data set” |
| 8422 | E | | “Record length exceeds device limits” |
| 8423 | E | | “Record larger than current DASD extent” |
| 8424 | E | | “DASD device is output only” |
| 8425 | E | | “Unrecoverable input/output error ” “while processing tape labels” |
| 8426 | E | | “Unrecoverable input/output error ” “while positioning a tape volume” |
| 8427 | E | | “Tape processing had a block count mismatch” |
| 8428 | E | | “Unrecoverable input/output error ” “processing a DASD label or VTOC” |
| 8429 | E | | “Member name missing in ” “partitioned data set directory” |
| 8430 | E | | “Unrecoverable subsystem error” |
| 8431 | E | | “Tape density not supported on requested device” |
| 8432 | E | | “Tape device or volume already in use” |
| 8433 | E | | “Unrecoverable error” “during checkpoint processing” |
| 8434 | E | | “Unable to process next volume of the data set” |
| 8435 | E | | “Data set is already in use ” “as the OUTPUT destination” |
| 8436 | E | | “No volume serial specified” |
| 8437 | E | | “Volume label validation error” |
| 8438 | E | | “Unable to process volume label” |
| 8439 | E | | “Device unavailable or unusable” |
| 8440 | E | | “Volume sequence number exceeds volume count” |
| 8441 | E | | “Data set not defined to RACF” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8442 | E | | “Invalid PROTECT specification” |
| 8443 | E | | “Cannot output to concatenated partitioned data set” |
| 8444 | E | | “Invalid concatenated partitioned data set” |
| 8445 | E | | “ANSI/ISO/FIPS variable records are not ” “supported for partitioned data sets” |
| 8446 | E | | “Machine code printer control is invalid ” “for ANSI/ISO/FIPS data sets” |
| 8447 | E | | “DCB=BFTEK=A is required to process ANSI/ISO/FIPS” “spanned records” |
| 8448 | E | | “HDELETE request failed, ” “ARCHDEL macro return code %d” |
| 8448 | E | | “HRECALL request failed, ” “ARCHRCAL macro return code %d” |
| 8449 | E | | “HDELETE request failed, ““refer to MVS message ARC11%02.2dI” or “HRECALL request failed, ” “refer to MVS message ARC11%02.2dI” |
| 8450 | E | | “Uncatalog failed, CATALOG macro return code %d” |
| 8452 | E | | “CREATE option ‘%s’ is not supported” |
| 8454 | E | | “MVS catalog request failed, code (0x%08.8X)” |
| 8456 | I | I | “Catalog file sequence number ” “at maximum (9999)” |
| 8457 | I | | “Note: Referback to data set %s ignored” |
| 8458 | I | | “Note: Referback to DD name %s ignored” |
| 8459 | E | | “Error encountered during free of DD name %s” |
| 8460 | E | | “Unable to obtain a catalog search buffer” |
| 8460 | I | I | “Unable to obtain a catalog search buffer” |
| 8461 | E | | “Catalog search exceeded ” “available buffer size” |
| 8461 | I | I | “Catalog search exceeded ” “available buffer size” |
| 8462 | E | | “Too many catalog entries exist” “at level ‘%s’” |

| Table 5. Error Messages | | | |
|---|---|---|---|
| **Code** | **Severity** | **Comment** | **Text** |
| 8463 | E | | “HSM request failed for data set %s” |
| 8464 | E | | “Missing file name for HSM request” |
| 8465 | E | | “Missing data set name for HSM request” |
| 8466 | E | | “Unsupported HSM request ‘%s’” |
| 8602 | E | | “Invalid TSO prefix ‘%s’” |
| 8605 | E | I | “LOCAL_VALID: bad keyword ‘%s’” |
| 8606 | E | A | “Invalid SPACE value ‘%s’” |
| 8903 | E | A | “Remote BATCH is not supported” |
| 9000 | E | | “Missing source file name” |
| 9001 | E | A | “Too many cataloged data sets matched the wildcard ” “pattern” |
| 9003 | E | A | “Wildcard processing error - code (%d)” |
| 9004 | E | A D | “Invalid data set name or wildcard pattern” |
| 9005 | E | A | “No cataloged data sets matched the wildcard ” “specification” |
| 9006 | E | A | “Wildcarding is not supported ” “in the first data set qualifier” |
| 9007 | E | | “Unable to obtain %d bytes of dynamic memory” |
| 9008 | E | | “Error occurred while processing a ” “partitioned data set directory” |
| 9009 | E | | “Error occurred during catalog search” |
| 9011 | E | | “Wildcard buffer size: %d” |
| 9999 | E | | “Pipi_init Failure rc=%i” |
| 9999 | I | | “ENVPUT TCPIPS Failed” |

# Additional Descriptions

This list provides additional descriptions for some NetEx/eFT messages. The descriptions below expand on the information in the table.

**UA-302 Overflow of NNN byte environment buffer**

**Severity:** Error

**Explanation:** User data is stored in fixed length environment buffers and the string that was to be added caused the environment buffer to overflow.

**UA-303 Failed to add 'SSS=SSS'**

**Severity:** Error

**Explanation:** The variable name and its definition (truncated to 15 characters) will be displayed. SSS=SSS represents the variable addition to the environment that would not fit. To remedy this problem, reduce the length of the name or the size of the description or if attempting to add to the GLOBAL environment, use the -GLOBAL switch when invoking NetEx/eFT to increase the GLOBAL environment.

**UA-501 Protocol error - expected [CC] - got [CC]**

**Severity:** Error

**Explanation:** The protocol type in NetEx/eFT did not match the protocol type of the remote host. The probable cause is either a network interruption or a revision-level incompatibility between the Initiator and the Responder.

**UA-4106 The requested blocksize NNN was reduced to NNN**

**Severity:** Informational

**Explanation:** The reduction (and resultant message) will only occur during the connect process. First, the local NETEX and remote NETEX perform a block size negotiation, and then there is a secondary block size negotiation between the NetEx/eFT Responder and Initiator. During negotiation the requested block size gets sent to the remote host and a negotiated block size gets returned. The negotiated block size is always the smaller of the two hosts. When connecting to a version 1 Responder, this message may occur erroneously during a CONNECT. The block size is renegotiated during a version 1 file transfer.

**UA-4109 There were NNN CONNECT records ignored**

**Severity:** Warning

**Explanation:** The records that are ignored are typically records coming from a newer release of the Responder than the Initiator. In this case the Responder sends more CONNECT information than the Initiator knows how to use. The message provides a warning that the connection may not support all of the functionality offered by the Responder.

**UA-4127 The MESSAGE stack is empty**

**Severity:** Error

**Explanation:** An error has occurred, but there is no message associated with the error.

**UA-4131 Failed to establish secondary NETEX connection**

**Severity:** Error

**Explanation:** Some NETEX Responders need a second connection to perform file transfers. There is likely to be a NETEX error (e.g. too many sessions); check the NETEX message if one is provided and retry. This error could occur as a result of a timeout or because of a revision-level incompatibility between the Initiator and the Responder.

**UA-4132 Restricted command in server startup file**

**Severity:** Error

**Explanation:** For security reasons, NetEx/eFT server startup files may not contain any of the following commands: CONNECT, DISCONNECT, LOCAL, RECEIVE, REMOTE, or SEND. These commands may not be embedded within NetEx/eFT aliases.

**UA-4133 ProdConfRead Error**

**Severity:** Error

**Explanation:** This message is displayed when there are errors in reading the PRODCONF file. The specific message text will vary depending on the cause of the problem. The messages that could be reported are:

- "Failed to open product config file 'SSSSSS'" where SSSSSS is the name of the prodconf file the license verification program attempted to access.
- "No data in file 'SSSSSS'" indicates that the prodconf file, indicated by SSSSSS, is empty.
- "Missing LICPATH value in file 'SSSSSS'" indicates that the prodconf file does not contain a value for the LICPATH parameter.

**UA-4134 EFTUTL1: CPU Serial Number could not be obtained.**

**Severity:** Error

**Explanation:** The license verification routine was unable to read the CPU serial number.

**UA-4135 License Verify Error**

**Severity:** Error

**Explanation:** This message is displayed when there are errors in reading the license keys file or the contents of that file. The specific message text will vary depending on the cause of the problem. The messages that could be reported are:

- "Failed to open NESikeys file 'SSSSSS'" indicating that keys file SSSSSS does not exist or cannot be accessed for other reasons.
- "Product EFT213 not licensed to run on this platform" indicates that a valid key for the referenced product does not exist.
- License keys may be invalid due to:
  - License expiration
  - Incorrect LPAR specified when requesting the License Key
  - Incorrect S/N specified when requesting the License Key
  - No Key in the keys file

- Old or corrupt key

**UA-4136 License Expiring Warning**

**Severity:** Warning

**Explanation:** This message is displayed when the current license key is close to expiring. The expiration date can also be found by performing a 'Show local/remote'. A new license key should be obtained and installed prior to the license expiration to prevent a disruption in ser-vice.

**UA-4137 LicenseVerifyCapability Error**

**Severity:** Error

**Explanation:** This message is displayed when there is an incompatibility between the NetEx/eFT features configured and the features enabled by the software key. The actual text of the message will vary depending on the reason for the incompatibility. Possible messages include:

- "Product License protocol CAPability is incorrect." This message indicates that NetEx/eFT is attempting to run over a protocol (NetEx versus TCP/IP) that it is not licensed to use.

**UA-4138 LicenseVerifyCapability Warning**

**Severity:** Warning

**Explanation:** License Capabilities Verify warning (reserved for future use).

**UA-4139 License Expired and Product will soon be non-operational**

**Severity:** Warning

**Explanation:** NetEx/eFT License keys contain expiration dates. This license key contains a grace period, which is allowing you to run past your term date. When the grace period is reached, the product will become not operational.

**UA-4140 License Expired and Product is not operational**

**Severity:** Error

**Explanation:** NetEx/eFT License keys contain expiration dates. When the expiration date is reached, the product will become not operational.

**UA-4501 Nested (or recursive) input/alias limit of NNN exceeded**

**Severity:** Error

**Explanation:** NetEx/eFT restricts the number of times an input script or alias can call itself or another script/alias; the current limit is ten levels. This error can also be a result of a user failing to escape alias processing (using the '!' escape character) when redefining a NetEx/eFT command as an alias within a multicommand alias.

**UA-4504 Bad output FORMAT definition - reset to default**

**Severity:** Error

**Explanation:** A user can redefine the format of error messages by using the SET OUTPUT FORMAT command. This message results when the new definition does not begin with '{}' to disable string substitution.

**UA-4505 Input request (NNN byte maximum) failed**

**Severity:** Error

**Explanation:** NetEx/eFT provides a buffer for holding a multiline command or alias; this error occurs when that buffer is exceeded. If a large command or alias is required, it should be defined as an Input Script.

**UA-4601 Variable 'SSS' contains invalid characters**

**Severity:** Warning

**Explanation:** A variable name was created that contains invalid characters; SSS represents the variable name. Valid characters are alphanumeric 'A'.. 'Z', '0'.. '9'. It is recommended for future compatibility that variable names and alias names begin with an alpha character 'A'.. 'Z'.

**UA-4701 Recursive alias 'SSS'**

**Severity:** Warning

**Explanation:** A warning message resulted when NetEx/eFT attempted to execute a single line alias that was recursive (it calls itself). A common cause of this error is executing an alias that calls an alias that calls the first alias back again.

**UA-4704 Use SET LOCAL/REMOTE to modify SSS qualifier 'SSS'**

**Severity:** Warning

**Explanation:** Some LOCAL and REMOTE qualifiers cannot be modified with some NetEx/eFT commands. This error occurs if a user attempts to modify either current directory by means of a -DIR switch on a SEND, RECEIVE, LOCAL, or REMOTE command line.

**UA-4709 Command token is greater than NNN characters**

**Severity:** Warning

**Explanation:** A token is a sequence of characters separated by either blanks, tabs, end-of-line, or any combination thereof. The token cannot exceed NNN length. Note that a token, enclosed in quotes, can include spaces.

**UA-4804 MAXRECORD (NNN+NNN) too large for BLOCKSIZE (NNN)**

**Severity:** Error

**Explanation:** MAXRECORD plus the record header size must be less than or equal to the BLOCKSIZE negotiated at connect time. To correct the error, reduce the MAXRECORD qualifier, reconnect with a larger BLOCKSIZE, or enable the PARTialrecord qualifier.

**UA-4809 Sequence error (NNN vs. NNN) in record RECEIVE**

**Severity:** Error

**Explanation:** NetEx/eFT has a sequence number associated with each record in a RECORD MODE file transfer. A sequence error is probably caused by a network interruption.

**UA-5120 The MESSAGE stack is empty**

**Severity:** Error

**Explanation:** Refer to previous identical message: UA-4127

**UA-5206 Use SET LOCAL/REMOTE to modify SSS qualifier 'SSS'**

**Severity:** Warning

**Explanation:** Refer to previous identical message: UA-4704

**UA-5304 Invalid ARCHIVE file format [SSS]**

**Severity:** Error

**Explanation:** This error results from trying to use the RESTORE MODE to SEND or RECEIVE a file that was not created by a BACKUP MODE transfer, or to use COPY MODE to SEND or RECEIVE a file to or from a host that is a different type from the local host (i.e. not peer-to-peer).

**UA-5305 Invalid ARCHIVE block length (NNN)**

**Severity:** Error

**Explanation:** This error results from trying a RESTORE or COPY MODE file transfer on an incompatible HOSTTYPE or ARCHIVE file.

**UA-5306 Incomplete ARCHIVE file - missing end-of-file**

**Severity:** Error

**Explanation:** This error results from trying a RESTORE mode file transfer on a container file that (for some reason) is not complete. The most likely reason is that the BACKUP mode transfer that created the file was aborted, leaving a partial file with missing data and no Archive end-of-file mark.

**UA-5401 SSS more than NNN levels of nested strings**

**Severity:** Warning

**Explanation:** This warning occurs with string substitution. If the nesting level is more than NNN, this warning results. (i.e., if NNN is 8, then {{{{{{{{{password}}}}}}}}} causes a warning.)

**UA-5406 Empty string substitution**

**Severity:** Warning

**Explanation:** This warning results when NetEx/eFT is unable to find an alphanumeric string (string variable or function) where one was expected. This is generally due to a syntax problem caused by a missing parameter to a string function or a missing function name itself. Make sure that a string substitution does not result in a null string. For example, placing too many, or unnecessary brackets '{' '}' around a variable or argument will cause this warning condition.

**UA-5603 Character code cannot be translated**

**Severity:** Warning

**Explanation:** This warning results from the TRANSLATE command to define character translations. Characters that cannot be redefined are Uppercase alpha ("A".. "Z"), digits ("0".. "9"), space(" "), equal ("="), and null ("").

**EFT213-833 %s service not configured. Using program defaults**

**Severity:** Informational

**Explanation:** The "getservbyname" system macro failed to retrieve the service name from the TCP.ETC.SERVICES file. The program defaults are used:

```
"EFT",          6900,           /* EFT */
"EFTS",         6910,           /* EFT Secure */
"USER",         6900,           /* EFT */
/*  "CAMLTS",       6910,     */    /* CAM Local Tape Server */
"TEFT",         6920,           /* EFT Full Screen */
"TUSER",        6920,           /* EFT Full Screen */
"EFTGATE",      6930,           /* EFT-Gate */
"USERGATE",     6930,           /* EFT-Gate */
"SIHELPER",     6940,           /* SI Helper */
/*  "CAM",          6950,     */    /* CAM Graphical Server */
/*  "CAMSDB",       6960,     */    /* CAM Secure Database */
/*  "CAMRM",        6970,     */    /* CAM Resource Manager */
/*  "CAMNS",        6980,     */    /* CAM Directory Name Server */
/*  "CAMCMD",       6990,     */    /* CAM Command Server */
0,              0               /* end-of-table */
```

**EFT213-2002 Data checksum (CRC) error at block NNN**

**Severity:** Error

**Explanation:** File transfer with -CRC ON detected a checksum error in block number SSS. The 32-bit checksum is computed by the source host and verified by the destination host. The verification failed due to some network interruption. The file transfer is aborted and the destination file may not be complete. Correct the problem and retry the command.

**EFT213-2004 Sequence number error at block NNN**

**Severity:** Error

**Explanation:** File transfer with -CRC ON detected a missing block at block number SSS. The block sequence number is computed by the source host and verified by the destination host. The verification failed due to some network interruption. The file transfer is aborted and the destination file may not be complete. Correct the problem and retry the command.

**EFT213-2101 Failed to allocate NNN bytes of dynamic memory.**

**Severity:** Error

**Explanation:** File transfer was unable to obtain a sufficient amount of memory for buffers. The file transfer is aborted. Increase the amount of z/OS virtual storage (REGION) available or reduce the number of active connections, and retry the command.

**EFT213-3100 Support documentation messages**

**Severity:** Informational

**Explanation:** Additional information that may be needed by customer support to diagnose or resolve an issue.

**EFT213-3101 Fipslvl can only be set once, and turned off.**

**Severity:** Error

**Explanation:** EFT must be terminated and restarted to change the FIPSLVL once it has been set.

**EFT213-3102** The value for certdb was not found in the GSK database

**Severity:** Error

**Explanation:** The value for certdb was incorrect or missing from the GSK database. Correct the value.

**EFT213-3103** The value for certpw did not match the password for the database.

**Severity:** Error

**Explanation:** Specify the read password value for the database.

**EFT213-3104** The dataset specified by certdb was not found

**Severity:** Error

**Explanation:** The protocol type in NetEx/eFT did not match the protocol type of the remote host. The probable cause is either a network interruption or a revision-level incompatibility between the Initiator and the Responder.

**EFT213-3105 Connection was reset by peer. Check for -secure -nosecure settings**

**Severity:** Error

**Explanation:** The remote side closed the connection. This may be caused by one side setting **-secure** and the other side setting **-nosecure**.

**EFT213-8001 Keyboard interrupt**

**Severity:** Error

**Explanation:** The user issued a TSO/E attention request. The current NetEx/eFT command is aborted.

**EFT213-8201 SSS command execution is not supported**

**Severity:** Error

**Explanation:** Command execution is not supported for the specified host.

**EFT213-8202 Command input error**

**Severity:** Error

**Explanation:** The local command exceeded the command input buffer length. Local command execution is aborted.

**EFT213-8203 Note: SSS TSOPREFIX may not match DIRECTORY**

**Severity:** Informational

**Explanation:** The user issued a TSO/E PROFILE command to change the TSO/E prefix. The NetEx/eFT DIRECTORY qualifier may not match the new TSO/E prefix. Use the SET LOCAL DIRECTORY or SET REMOTE DIRECTORY command to reset both the DIRECTORY and TSO/E prefix.

**EFT213-8300 Failed to access data set SSS**

**Severity:** Error

**Explanation:** NetEx/eFT encountered an error while trying to access the specified data set. Correct the problem(s) in any following messages and retry the command.

**EFT213-8301 Failed to create data set SSS**

**Severity:** Error

**Explanation:** NetEx/eFT encountered an error while trying to create the specified data set. Correct the problem(s) in any following messages and retry the command.

**EFT213-8302 Failed to delete data set SSS**

**Severity:** Error

**Explanation:** NetEx/eFT encountered an error while trying to delete the specified data set. Correct the problem(s) in any following messages and retry the command.

**EFT213-8303 No data set name or DD name was specified**

**Severity:** Error

**Explanation:** The specified source file name is missing or invalid. Verify the source file name is correct and retry the command.

**EFT213-8304 Data set not in catalog**

**Severity:** Error

**Explanation:** The specified data set was not found in the z/OS catalog. Verify the data set name is correct and retry the command.

**EFT213-8305 Cataloged data set not found on volume SSS**

**Severity:** Error

**Explanation:** The specified data set was found in the z/OS catalog, but not on the cataloged volume. Verify the z/OS catalog entry is correct and retry the command.

**EFT213-8306 Data set not found on volume SSS**

**Severity:** Error

**Explanation:** The specified data set was not found on the specified volume. Verify the data set and volume names are correct and retry the command.

**EFT213-8307 DD name SSS is not allocated**

**Severity:** Error

**Explanation:** The specified DD name was not previously allocated. Verify the DD name is correct and allocated, and retry the command.

**EFT213-8308 Cannot process concatenated DD name SSS**

**Severity:** Error

**Explanation:** The specified DD name points to a list of concatenated data sets. NetEx/eFT for IBM z/OS does not support concatenated data sets.

**EFT213-8309 Volume SSS is not available**

**Severity:** Error

**Explanation:** The volume name specified or obtained from the z/OS catalog is not online or is unavailable. Verify the volume name is correct, the volume is online and available, and retry the command.

**EFT213-8311 Dynamic allocation failed, error code (NNN), reason code (NNN)**

**Severity:** Error

**Explanation:** z/OS data set dynamic allocation failed. Correct the problem(s) in any following z/OS messages and retry the command. If there are no following z/OS messages, refer to the IBM publication "z/OS System Programming Library: System Macros and Facilities" for an explanation of the error and reason codes.

**EFT213-8312 Unrecoverable input/output error**

**Severity:** Error

**Explanation:** An unrecoverable input or output error occurred while processing a data set. You may need to contact your site administrator to report and correct the problem.

**EFT213-8313 Unrecoverable input/output error reading partitioned data set directory**

**Severity:** Error

**Explanation:** An unrecoverable input or output error occurred while processing a partitioned data set directory. You may need to contact your site administrator to report and correct the problem.

**EFT213-8314 IBM/MVS ABEND code SNNN-NN**

**Severity:** Error

**Explanation:** An z/OS system ABEND (abnormal termination) was encountered while processing a data set. Refer to the IBM publication "z/OS Message Library: System Codes" for explanation of the ABEND and reason code. If no other NetEx/eFT messages were produced, you may need to contact your site administrator to report and correct the problem.

**EFT213-8324 Data set exists and -CREATE NEW was specified**

**Severity:** Error

**Explanation:** You cannot overwrite an existing data set with the CREATE qualifier set to NEW. Retry the command with a different CREATE option.

**EFT213-8325 Member SSS exists and -CREATE NEW was specified**

**Severity:** Error

**Explanation:** You cannot overwrite an existing partitioned data set member with the CREATE qualifier set to NEW. Retry the command with a different CREATE option.

**EFT213-8326 Member SSS does not exist**

**Severity:** Error

**Explanation:** The specified member does not exist in the partitioned data set. Verify the data set and member names are correct and retry the command.

**EFT213-8327 Text record longer than NNN bytes**

**Severity:** Error

**Explanation:** File transfer detected a record longer than the maximum length allowed for the data set. The file transfer is aborted and the destination file may not be complete.

**EFT213-8328 Data set contains noncharacter data**

**Severity:** Error

**Explanation:** A data set being used by the INPUT command or character mode file transfer contains noncharacter data. The input file is flushed or file transfer is aborted and the destination file may not be complete.

**EFT213-8329 Character input longer than NNN bytes**

**Severity:** Error

**Explanation:** Character input exceeded the maximum length allowed. The input is ignored, the input file is flushed, or the file transfer is aborted and the destination file may not be complete.

**EFT213-8330 DESTINATION name contains too many wildcard characters**

**Severity:** Error

**Explanation:** The destination file name contains more than two wildcard characters. Verify the destination file name is correct and retry the command.

**EFT213-8331 SPACE value required when creating a data set**

**Severity:** Error

**Explanation:** The default SPACE qualifier value was not provided, no space value was provided on the SEND or RECEIVE command, and the source file did not have an estimated space amount. Retry the command with a valid space value.

**EFT213-8332 CREATE APPEND is not valid for a partitioned data set**

**Severity:** Error

**Explanation:** z/OS does not support appending to a partitioned data set member. Retry the command with a different CREATE option.

**EFT213-8333 CREATE APPEND is not valid for a DD name**

**Severity:** Error

**Explanation:** NetEx/eFT does not support CREATE APPEND for DD name file transfers. Retry the command with a different CREATE option.

**EFT213-8334 CREATE DELETE is not valid for a partitioned data set**

**Severity:** Error

**Explanation:** NetEx/eFT does not support CREATE DELETE for partitioned data sets. Retry the command with a different CREATE option.

**EFT213-8335 CREATE DELETE is not valid for a DD name**

**Severity:** Error

**Explanation:** NetEx/eFT does not support CREATE DELETE for DD name file transfers. Retry the command with a different create option.

**EFT213-8336 Password required**

**Severity:** Error

**Explanation:** The specified data set is password protected and no password was provided on the SEND or RECEIVE command. Retry the command with a valid data set password.

**EFT213-8337 Record longer than maximum allowed of NNN**

**Severity:** Error

**Explanation:** A data set being used by the INPUT command or file transfer contained a record longer than the maximum length allowed. The input file is flushed, or file transfer is aborted and the destination file may not be complete.

**EFT213-8338 Error encountered while reading from data set**

**Severity:** Error

**Explanation:** An unrecoverable error was encountered while reading from a data set. Correct the problem(s) in any following messages and retry the command.

**EFT213-8339 Error encountered while writing to data set**

**Severity:** Error

**Explanation:** An unrecoverable error was encountered while writing to a data set. Correct the problem(s) in any following messages and retry the command.

**EFT213-8340 Error encountered while writing to output file - SSS**

**Severity:** Error

**Explanation:** An unrecoverable error was encountered while writing to an output file (SYSPRINT, etc.). Correct the problem and retry the command.

**EFT213-8342 RECFORMAT (SSS) is not supported for CHARACTER mode transfers**

**Severity:** Error

**Explanation:** The specified record format is not supported for character mode file transfer. Retry the command with a different record format or transfer mode.

**EFT213-8343 RECFORMAT (SSS) is not supported for RECORD mode transfers**

**Severity:** Error

**Explanation:** The specified record format is not supported for record mode file transfer. Retry the command with a different record format or transfer mode.

**EFT213-8344 RECFORMAT (SSS) is not supported for STREAM mode transfers**

**Severity:** Error

**Explanation:** The specified record format is not supported for stream mode file transfer. Retry the command with a different record format or transfer mode.

**EFT213-8345 Data set has no RECFORMAT defined**

**Severity:** Error

**Explanation:** The data set cannot be used because no record format was provided. Retry the command with a valid record format.

**EFT213-8346 RECFORMAT (SSS) is not currently supported**

**Severity:** Error

**Explanation:** The data set cannot be used because the specified record format is not supported by NetEx/eFT. Retry the command with a supported record format.

**EFT213-8347 Record length (NNN) exceeds maximum record length (NNN)**

**Severity:** Error

**Explanation:** The specified record length is larger than the maximum record length. Retry the command with a smaller record length.

**EFT213-8348 RECLENGTH (NNN) is out of range (NNN - NNN)**

**Severity:** Error

**Explanation:** The specified record length is not within the range shown. Retry the command with a record length within the range shown.

**EFT213-8349 BLOCKSIZE (NNN) is out of range (NNN - NNN)**

**Severity:** Error

**Explanation:** The specified block size is not within the range shown. Retry the command with a block size within the range shown.

**EFT213-8350 BLOCKSIZE (NNN) must be a multiple of RECLENGTH (NNN) for RECFORMAT (SSS)**

**Severity:** Error

**Explanation:** The specified block size must be a multiple of the record length for the record format shown. Retry the command with a valid block size and record length.

**EFT213-8351 BLOCKSIZE (NNN) must equal RECLENGTH (NNN) for RECFORMAT (SSS)**

**Severity:** Error

**Explanation:** The specified block size must equal the record length for the record format shown. Retry the command with a valid block size and record length.

**EFT213-8352 Data set has no organization defined**

**Severity:** Error

**Explanation:** The data set cannot be used because no data set organization was provided. Retry the command with a valid data set organization.

**EFT213-8353 Data set organization (SSS) is not currently supported**

**Severity:** Error

**Explanation:** The specified data set organization is not supported. Retry the command with a different data set organization or convert the data set to a supported data set organization.

**EFT213-8354 Spanned records are not supported for partitioned data sets**

**Severity:** Error

**Explanation:** z/OS does not support variable length spanned records for partitioned data sets. Retry the command with a different record format.

**EFT213-8355 Standard records are not supported for partitioned data sets**

**Severity:** Error

**Explanation:** z/OS does not support fixed length standard records for partitioned data sets. Retry the command with a different record format.

**EFT213-8357 Member name invalid for a sequential data set**

**Severity:** Error

**Explanation:** The specified data set is a sequential data set and a member name cannot be provided. Retry the command without a member name.

**EFT213-8358 Member name omitted for a partitioned data set**

**Severity:** Error

**Explanation:** The specified data set is a partitioned data set and no member name was provided. Retry the command with a valid member name.

**EFT213-8359 Zero length record invalid for a data set with printer control**

**Severity:** Error

**Explanation:** The specified data set requires ANSI or machine code printer control characters as the first character of every record. File transfer detected a zero-length record which is not valid for this type of data set. Retry the command with a different data set or record format. The output is aborted and the destination file may be incomplete.

**EFT213-8360 Primary filled and no secondary SPACE available**

**Severity:** Error

**Explanation:** The output data set filled the available primary space and no secondary space amount was provided. The output is aborted and the data set will not be complete.

**EFT213-8361 Space on the current volume filled and no more volumes available**

**Severity:** Error

**Explanation:** The output data set filled the available primary and secondary space on the current volume and no more volumes are available. The output is aborted and the data set will not be complete.

**EFT213-8362 Space on the current volume filled and next volume unavailable**

**Severity:** Error

**Explanation:** The output data set filled the available primary and secondary space on the current volume and next volume requested is unavailable. The output is aborted and the data set will not be complete.

**EFT213-8363 Space on the current volume filled and next volume has too many users**

**Severity:** Error

**Explanation:** The output data set filled the available primary and secondary space on the current volume and next volume already has the maximum number of active users assigned. The output is aborted and the data set will not be complete.

**EFT213-8364 Space on the current volume filled and installation exit rejected next volume**

**Severity:** Error

**Explanation:** The output data set filled the available primary and secondary space on the current volume and an z/OS installation exit has denied access to the next volume. The output is aborted and the data set will not be complete. You may need to contact your site administrator to find why access to the next volume was denied.

**EFT213-8365 (SSS) is longer that maximum of NNN character(s)**

**Severity:** Error

**Explanation:** The specified name or qualifier is longer than the maximum length allowed. Retry the command with a valid name or qualifier.

**EFT213-8366 Invalid or unexpected character 'C' found in SSS**

**Severity:** Error

**Explanation:** The specified name or qualifier contains an invalid or unexpected character. Retry the command with a valid name or qualifier.

**EFT213-8367 Requested transfer mode is not supported**

**Severity:** Error

**Explanation:** The requested file transfer mode is not supported by IBM/z/OS NetEx/eFT. Retry the command with a different transfer mode.

**EFT213-8368 Unmatched apostrophes in data set name**

**Severity:** Error

**Explanation:** The data set name contains unmatched quotes (apostrophes). Retry the command with a valid data set name.

**EFT213-8369 Unmatched parenthesis in data set name**

**Severity:** Error

**Explanation:** The data set name contains an unmatched set of parentheses. Retry the command with a valid data set name.

**EFT213-8370 Bad qualifier in data set name**

**Severity:** Error

**Explanation:** The data set name contains an invalid data set name qualifier. Retry the command with a valid data set name.

**EFT213-8371 Partitioned data set directory is full**

**Severity:** Error

**Explanation:** The partitioned data set directory is full, and the requested member cannot be added. Delete any unneeded members from the partitioned data set or expand the partitioned data set directory, and retry the command.

**EFT213-8373 Unknown archive mode tag field [SSS]**

**Severity:** Error

**Explanation:** Archive mode file transfer detected a record with an unknown archive mode tag. The file transfer is aborted and the destination file may not be complete.

**EFT213-8374 Invalid ARCHIVE tag field [SSS]**

**Severity:** Error

**Explanation:** Archive mode file transfer detected a record with an invalid archive mode tag field. The file transfer is aborted and the destination file may not be complete.

**EFT213-8375 Bad ARCHIVE record length NNN**

**Severity:** Error

**Explanation:** Archive mode file transfer detected a record with an invalid record length. The file transfer is aborted and the destination file may not be complete.

**EFT213-8376 Note: NNN record(s) truncated at NNN characters**

**Severity:** Informational

**Explanation:** A file transfer with the WRAP qualifier set to OFF detected one or more records that were longer than the destination file maximum record length. The detected records were truncated at the maximum length and the remainder of the original record was discarded.

**EFT213-8377 Note: NNN record(s) wrapped at NNN characters**

**Severity:** Informational

**Explanation:** A file transfer with the WRAP qualifier set to ON detected one or more records that were longer than the destination file maximum record length. The detected records were truncated at the maximum length and the remainder of the original record was continued on the next record.

**EFT213-8378 Note: NNN record(s) padded to NNN characters**

**Severity:** Informational

**Explanation:** A file transfer to a data set with fixed length records detected one or more records shorter than the fixed record length. The detected records were padded to meet the fixed record length.

**EFT213-8379 Tape label data set name mismatch**

**Severity:** Error

**Explanation:** The specified data set name did not match the data set name on the specified tape volume label. Verify the data set name and tape label are correct, then retry the command.

**EFT213-8380 Invalid password**

**Severity:** Error

**Explanation:** The specified data set password is invalid or incorrect. Retry the command with a valid data set password.

**EFT213-8381 You are not authorized to access this data set**

**Severity:** Error

**Explanation:** A security system at your installation denied access to the specified data set. Retry the command with a different data set name or password, and if the problem persists, you may need to contact your site administrator to find why access to the data set was denied.

**EFT213-8603 Invalid directory 'SSS'**

**Severity:** Error

**Explanation:** The specified directory is not valid. Retry the command with a valid directory.

**EFT213-8604 Failed to get working directory - SSS**

**Severity:** Error

**Explanation:** NetEx/eFT could not obtain access to the specified directory. Retry the command with a valid directory.

**EFT213-8606 Invalid SPACE value 'SSS'**

**Severity:** Error

**Explanation:** The SPACE qualifier value is not valid. Retry the command with a valid space value.

**EFT213-8701 Missing DESTINATION specifier**

**Severity:** Error

**Explanation:** The file transfer destination file name is missing or invalid. Retry the command with a valid destination file name.

**EFT213-8702 Piped command failed with an exit status of SSS**

**Severity:** Informational

**Explanation:** The piped command for file transfer may not have completed successfully. Verify the piped command worked successfully and retry the command if necessary.

**EFT213-8903 Remote BATCH is not supported**

**Severity:** Error

**Explanation:** The NetEx/eFT remote batch facility is not implemented for IBM/z/OS NetEx/eFT. Use the TSO/E SUBMIT command instead.

**EFT213-9001 Missing source data set name**

**Severity:** Error

**Explanation:** The file transfer source file name is missing or invalid. Retry the command with a valid source file name.

**EFT213-9002 Too many cataloged data sets matched the wildcard pattern**

**Severity:** Error

**Explanation:** The number of data sets and/or partitioned data set members exceeded the maximum allowed for a wildcard file transfer. Retry the command with a wildcard pattern that will not select as many files.

**EFT213-9003 Wildcard processing error - code (NNN)**

**Severity:** Error

**Explanation:** Wildcard processing failed due to an internal error. Retry the command and if the problem persists, you may need to contact your site administrator to report and correct the problem.

**EFT213-9004 Invalid data set name or wildcard pattern**

**Severity:** Error

**Explanation:** The data set name or wildcard pattern is invalid. Retry the command with a valid data set name or wildcard pattern.

**EFT213-9005 No cataloged data sets matched the wildcard pattern**

**Severity:** Error

**Explanation:** A search of the z/OS catalog did not find any data set names that matched the wildcard pattern. Retry the command with a different data set name or pattern.

**EFT213-9006 Wildcarding is not supported in the first data set qualifier**

**Severity:** Error

**Explanation:** z/OS does not allow wildcarding in the first qualifier of a data set name. Retry the command with a valid data set name or pattern.

**MUX-2501 Invalid TCP service name 'nnnn'**

**Severity:** Error

**Explanation:** The specified service name was not properly configured in the TCPIP.ETC.SERVICES dataset. This may also be caused by using a secondary TCPIP stack and not having the proper SYSTCPD DD card specified. A secondary cause could be someone editing this dataset when the MUX was checking for the service names. Restarting the EFTSERVER after the user is finished editing the dataset will correct this problem.

**MUX-4001 Invalid CONTROL password**

**Severity:** Warning

**Explanation:** The specified Multiplex Server control password is invalid or incorrect. Verify the control password is correct and retry the command.

**MUX-4002 Multiplexed Server stopped**

**Severity:** Warning

**Explanation:** The Multiplex Server has accepted the stop request.

**MUX-4003 Trace flag settings: SSS**

**Severity:** Warning

**Explanation:** The new Multiplex Server trace flag settings are as shown.

**MUX-4004 INFO not implemented**

**Severity:** Warning

**Explanation:** The Multiplex Server information facility is not implemented for IBM/z/OS NetEx/eFT.

**MUX-4005 Invalid CONTROL request 'SSS'**

**Severity:** Warning

**Explanation:** The specified Multiplex Server control request was invalid. Retry the command with a valid control request.

**MUX-4011 Sorry - SSS session limit exceeded (NNN active)**

**Severity:** Error

**Explanation:** The number of remote NetEx/eFT users is already at the maximum allowed. Retry the command at a later time.

**MUX-4012 Failed to login as user 'SSS'**

**Severity:** Error

**Explanation:** The remote NetEx/eFT login did not complete successfully. Correct the problem(s) in any following messages and retry the command.

**MUX-4014 A second NETEX connection is required**

**Severity:** Error

**Explanation:** A secondary NETEX connection is required for this command but could not be obtained. Verify a secondary NETEX connection is available and retry the command.

**MUX-4015 Incompatible NetEx/eFT version for file transfer**

**Severity:** Error

**Explanation:** The version of NetEx/eFT on the host you are currently connected to does not support file transfer with an z/OS host. File transfer cannot be accomplished unless a more recent version of NetEx/eFT is installed on the connected host.

**MUX-4016 Invalid code table length NNN**

**Severity:** Error

**Explanation:** When NetEx/eFT attempted to obtain the NETEX code conversion table, the table length returned was not 256 bytes. Correct the NETEX or hardware problem and retry the command.

**MUX-4017 Unsupported request code 'SSS' - no action taken**

**Severity:** Error

**Explanation:** The Multiplex Server detected an unsupported request code. Retry the command and if the problem persists, you may need to contact your site administrator to report and correct the problem.

**MUX-4018 Failed to invoke helper SSS**

**Severity:** Fatal

**Explanation:** The Multiplex Server issued the TSO/E CALL command to invoke the Helper program, but initial communication could not be established. Verify the Helper program name is correct in the Multiplex Server execution parameters and retry the command.

**MUX-4019 Timeout during helper initialization**

**Severity:** Fatal

**Explanation:** The Multiplex Server issued the TSO/E CALL command to invoke the Helper program and initial communication was established, but the Helper program did not complete communications within two minutes. This can sometimes occur on a very busy z/OS system. Retry the command and if the problem persists, you may need to contact your site administrator to report and correct the problem.

**MUX-4020 No client support for remote INPUT request**

**Severity:** Error

**Explanation:** The client program does not support remote interactive commands. The remote command can be retried if enough information is supplied on the command line to cause the command to become noninteractive.

**MUX-4021 Server RESULTS failed (type=SSS status=NNN)**

**Severity:** Error

**Explanation:** An error was detected during NetEx/eFT server processing. Retry the command and if the problem persists, you may need to contact your site administrator to report and correct the problem.

**MUX213-8001 Missing APPL_ID**

**Severity:** Error

**Explanation:** The TSO/E (TCAS) application name is missing in the Multiplex Server execution parameters. Correct the problem and restart the Multiplex Server.

**MUX213-8002 All connections to TSO are busy**

**Severity:** Error

**Explanation:** The number of remote NetEx/eFT sessions to TSO/E is already at the maximum allowed. Retry the command at a later time.

**MUX213-8003 Could not find PROCESS: SSS**

**Severity:** Error

**Explanation:** An error was detected during Multiplex Server login processing. Verify the TSO/E application name is the same in both the Multiplex Server execution parameters and the Multiplex Server ACF/VTAM configuration parameters. Correct the problem and restart the Multiplex Server.

**MUX213-8004 Login exceeded NNN second timeout**

**Severity:** Error

**Explanation:** TSO/E logon processing exceeded the maximum time allowed. This can be caused by an undetected TSO/E input prompt during logon or can sometimes occur on a very busy z/OS system. If the system is busy, specify a larger value for the **LOGTIMEOUT** keyword in the Multiplex Server execution parameters and restart the Multiplex Server. Correct the problem and retry the command.

**MUX213-8005 Missing USERNAME**

**Severity:** Error

**Explanation:** The TSO/E logon cannot complete because no username (TSO/E user ID) was supplied on the **CONNECT** command or defaulted in the Multiplex Server execution parameters. Retry the command with a valid username.

**MUX213-8006 No response supplied to a LOGIN prompt**

**Severity:** Error

**Explanation:** A TSO/E input prompt was detected during TSO/E logon processing. No response to the prompt was supplied on the CONNECT command or in the Multiplex Server execution parameters. Retry the command supplying a response to the prompt.

**MUX213-8008 Failed to connect to TSO**

**Severity:** Error

**Explanation:** TSO/E disconnected before logon could be completed. Verify the ACF/VTAM **LOGMODE** parameter is correct in the Multiplex Server ACF/VTAM configuration parameters. Correct the problem and restart the Multiplex Server.

**MUX213-8011 Failed to execute command: SSS**

**Severity:** Error

**Explanation:** The Multiplex Server could not pass a line of input to TSO/E. Retry the command and if the problem persists, you may need to contact your site administrator to report and correct the problem.

**MUX213-8012 VTAM connection lost**

**Severity:** Fatal

**Explanation:** The ACF/VTAM connection between the Multiplex Server and TSO/E has been lost. The NetEx/eFT session is disconnected. This can sometimes occur if the TSO/E user has been terminated after there has been no TSO/E activity for an extended period of time. Issue a new **CONNECT** command to reestablish the connection.