Connect:Enterprise UNIX®

Remote User's Guide

Version 2.4

Sterling Commerce
An IBM Company

Connect:Enterprise UNIX Remote User's Guide
Version 2.4

First Edition

STERLING COMMERCE SOFTWARE

*****TRADE SECRET NOTICE*****

Sterling Commerce, Inc.

4600 Lakehurst Court Dublin, OH 43016-2000 *
614/793-7000

Contents

Chapter 1 Accessing Connect:Enterprise 9

User Access to Connect:Enterprise 9
System Security 9
Logging On 10
Sending and Receiving Data 10
Customer-Supplied Security 10
Types of Communication Lines 10
Starting and Ending a Session 10
Host-Initiated Communications (Auto Connect) 11
Flags 11
Wildcard Syntax 13

Chapter 2 Standard FTP Commands 15

Requirements 15
Logging On to Connect:Enterprise 16
Changing Your Password 17
Logging Off FTP 18
cd Command 18
cd Command Format 18
Standard FTP cd Syntax 19
Standard FTP cd Examples 19
delete Command 20
del Command Format 21
Standard FTP delete Syntax 21
Standard FTP delete Examples 21
dir Command 22
dir Command Format 22
Output 23
Standard FTP dir Examples 24
get or recv Command 26
get or recv Command Format 26
Standard FTP get Syntax 26
Standard FTP get Examples 27
Unavailable Batches 28

mdelete Command 28
mdel Command Format 29
Standard FTP mdel Examples 29
mget Command 31
mget Command Format 31
Standard FTP mget Syntax 31
Standard FTP mget Examples 32
mput Command 34
mput Command Format 34
Standard FTP mput Syntax 34
Standard FTP mput Examples 34
put Command 36
put Command Format 36
Standard FTP put Syntax 36
Standard FTP put Examples 37
pwd Command 38
pwd Command Format 38
Example 38
rename Command 38
rename Command Format 38
Standard FTP put Syntax 39
Standard FTP rename Examples 39
Considerations When Using the del, mdel, get, and mget Commands 39

Chapter 3 FTP $$ Commands 43

Requirements 43
Logging On to Connect:Enterprise 44
Changing Your Password 45
Logging Off FTP 46
delete Command 46
del Command Format 46
$$ delete Syntax 47
$$ delete Examples 50
dir Command 50
dir Command Format 51
$$ dir Syntax 51
Output 55
$$ Syntax dir Examples 55
get Command 58
get Command Format 58
$$ get Syntax 58
$$ get Examples 62
Unavailable Batches 66
mget Command 66
mget Command Format 67
$$ mget Syntax 67
$$ mget Examples 70
put Command 73
put Command Format 73
$$ put Syntax 73
$$ put Examples 75

Invalid put Parameter Combinations 78
Determining Data Format Flag for Inbound Batches 78
Automatic Routing 79
Remote Site Script Usage 79

Chapter 4 SSH SFTP Commands 81

Requirements 81
Logging On to Connect:Enterprise 82
Logging Off SSH SFTP 82
cd Command 83
cd Command Format 83
Standard SSH SFTP cd Syntax 83
Standard SSH SFTP cd Examples 83
get Command 84
get Command Format 84
Standard SSH SFTP get Syntax 84
Standard SSH SFTP get Examples 84
Unavailable Batches 85
ls Command 85
ls Command Format 85
Standard SSH SFTP ls Syntax 85
Standard SSH SFTP ls Examples 86
put Command 86
put Command Format 86
Standard SSH SFTP put Syntax 86
Standard SSH SFTP put Examples 87
pwd Command 87
pwd Command Format 87
Example 87
rename Command 88
rename Command Format 88
Standard SSH SFTP rename Syntax 88
Standard SSH SFTP rename Examples 88
rm Command 88
rm Command Format 88
Standard SSH SFTP rm Syntax 89
Standard SSH SFTP rm Examples 89

Chapter 5 SSH SFTP $$ Commands 91

Logging Off SSH SFTP 91
rm Command 91
rm Command Format 92
$$ rm Syntax 92
$$ rm Examples 93
ls Command 93
ls Command Format 93
$$ ls Syntax 93
Output 96
$$ Syntax ls Examples 97

get Command 98
get Command Format 99
$$ get Syntax 100
$$ get Examples 103
Unavailable Batches 104
put Command 104
put Command Format 104
$$ put Syntax 104
$$ put Examples 106
Invalid put Parameter Combinations 107
Automatic Routing 108

Chapter 6 The SSH SCP Command 109

Adding a File with Standard Syntax 109
Add Examples 110
Extracting a File with Standard Syntax 111
Extract Examples 111
Adding a file with $$ Syntax 112
Available Keywords 112
Add Examples Using $$ 114
Extracting a file with $$ Syntax 114
Available Keywords 115
Extract Examples 117
Using an SCP Windows Client 118

Chapter 7 Exchanging Files Using WebDAV Protocol 119

Connecting to a Connect:Enterprise WebDAV Server 119
Connecting in Windows by Creating a Network Place 119
Connecting in Windows by Opening as a Web Folder 120
Connecting Using a Command Line WebDAV Client 120
Using Windows as a WebDAV Client 121
Viewing Contents of a Mailbox 121
Searching for a Batch 122
Available Keywords 123
Adding and Extracting Batches 123
Using a Command Line WebDAV Client 123
Viewing the Contents of a Mailbox 124
Viewing the Properties of a Batch 124
Extracting Batches 125
Available Keywords 126
Adding Batches 127
Available Keywords 127

Chapter 8 Async Protocol Commands 129

Requirements 129
Async Access Methods 130
Interactive Session Access Method 130
Noninteractive Access Method 130
Noninteractive Remote Sites Prefixing Data with $$ Cards 132

Logging On 132
Selecting an Async Transmission Protocol During an Interactive Session 133
Logging Off 133
$$ADD Command 134
$$ADD Command Format 134
Examples 136
Invalid $$ADD Parameter Combinations 138
Automatic Routing 138
$$DELETE Command 138
$$DELETE Command Format 139
Examples 142
$$DIRECTORY Command 143
$$DIRECTORY Command Format 144
Output 147
Examples 148
$$REQUEST Command 149
$$REQUEST Command Format 150
Required Parameters 151
Examples 155
Unavailable Batches 158

Chapter 9 Bisync Protocol Commands 159

Prerequisites 159
Sending a Command from a Remote Site to Connect:Enterprise 159
$$ADD Command 160
$$ADD Command Format 160
Optional Parameters 161
Invalid $$ADD Parameter Combinations 164
Automatic Routing 164
Examples 164
$$DELETE Command 166
$$DELETE Command Format 167
Examples 170
$$DIRECTORY Command 170
$$DIRECTORY Command Format 171
Required Parameter 171
Output 174
Examples 175
$$REQUEST Command 176
$$REQUEST Command Format 177
Required Parameters 178
Examples 183
Unavailable Batches 185

Index 187

Chapter 1

Accessing Connect:Enterprise

This chapter describes how to connect with Connect:Enterprise and describes security considerations.

User Access to Connect:Enterprise

Users of Connect:Enterprise have different methods of accessing the system. The reasons for this variety are:

- System security
- Types of communication lines

These two factors make each installation unique. The Connect:Enterprise system administrator at your local site can provide you with the specific information and procedures you need in order to access this product.

System Security

Security in the Connect:Enterprise system can be implemented at various points during processing. The manner in which Connect:Enterprise was installed determines the security requirements for your sessions. This section lists and describes all the possible levels of security available with Connect:Enterprise. Ensure that you have complete instructions from your Connect:Enterprise site administrator regarding what security is implemented for your sessions, what mailbox functionality is available to you, and what information you need to access the mailbox.

Security measures can affect operation when:

- You connect to Connect:Enterprise
- You log on to Connect:Enterprise
- You send or receive data
- You attempt to use specific command line utilities and/or API functions
- You access FTP services of Connect:Enterprise through several firewalls

Logging On

If you are accessing Connect:Enterprise from a TCP/IP network, you may encounter security measures when you log on to Connect:Enterprise from the FTP prompt for the first time. Also, many Bisync and Async protocols require you to log on with a password before you are allowed access to the mailbox. In all cases, your mailbox ID and password must be valid in order for Connect:Enterprise to accept the logon request.

Sending and Receiving Data

You may next encounter security measures when you send or receive data. If the Connect:Enterprise site uses the batch security feature, it neither accepts data from nor sends data to a remote site without a valid mailbox ID. Also, if the Connect:Enterprise site has implemented custom security using exits, you typically have to supply the PASSWORD parameter on your $$commands.

Customer-Supplied Security

Another area of security allows the Connect:Enterprise local site personnel to supply their own security measures. This security is unique for their installation and is not handled by the Connect:Enterprise security exit.

Types of Communication Lines

If you are communicating with Connect:Enterprise using Bisync or one of the Async protocols, the types of communication lines used can be a factor in how you initially connect to Connect:Enterprise. These sites use either switched or leased (nonswitched) lines.

A switched line is a temporary connection between two sites for the duration of the session only. A leased line is a permanent connection between sites that does not require a dialing to start communications.

Starting and Ending a Session

The procedures for starting and ending a session are different for each type of remote site. A typical procedure is:

1. Establish a connection.

   Your remote site may exist on a switched (dial up) or nonswitched (dedicated) line. You may have different procedures to follow when you first establish a connection with Connect:Enterprise. Your Connect:Enterprise system administrator should supply you with the proper procedures to establish a connection.

2. Log on (FTP, SSH SFTP, and interactive Async sites only).

   This procedure is done either manually or automatically and is remote-site dependent.

3. Ready to communicate with Connect:Enterprise.

   You can now enter the commands described in the following chapters.

4. Log off (FTP, SSH SFTP, and interactive Async sites only).

   See the procedures for logging off in Chapter 8, *Async Protocol Commands*, Chapter 2, *Standard FTP Commands*, Chapter 3, *FTP $$ Commands*, Chapter 4, *SSH SFTP Commands*, or Chapter 5, *SSH SFTP $$ Commands*.

Host-Initiated Communications (Auto Connect)

Connect:Enterprise has a function called Auto Connect that allows the local site to contact an unattended remote site. Using Auto Connect, Connect:Enterprise can both transmit to and collect data batches from remote sites.

For the Auto Connect function to work, the remote site must properly respond to Connect:Enterprise. Local site personnel can instruct you about preparing your site to respond to an Auto Connect from the repository.

Connect:Enterprise offers great flexibility in the way it can be configured to use the Auto Connect function. For example, Connect:Enterprise can be set up to transmit batches to your remote site at different times throughout the night. Your remote site terminal or computer is required to receive the batches whenever the host makes contact.

You can collect batches from the remote during the Auto Connect session. Because Auto Connect is often used with remote sites that can operate unattended, you probably need to set up your terminal to send batches when it is called by the host site.

You can also be instructed that you will receive a call from Connect:Enterprise at certain times of the day. You can then manually send batches to Connect:Enterprise, if instructed to do so.

Flags

This section contains a process flag table, a protocol flag table, and a data format flag table. The tables list all possible flag values.

The following table explains each process flag:

Process Flag	Description
A	Batch was added locally with the **cmuadd** command (for example, by the system administrator).
C	Batch was collected from a remote site.
D	Batch has been logically deleted.
E	Batch was extracted successfully at least once.
G	Batch was added with SSL.

Process Flag	Description
I	Collected batch is incomplete.
M	Batch may be transmitted multiple times.
N	Batch is permanently nontransmittable. Not available with the FLAGS parameter.
O	Batch is encrypted.
P	Online transmission is in progress. Not available with the FLAGS parameter.
R	Batch is requestable for remote connections or Auto Connects.
S	Indicates the batch is a LOG batch.
T	Batch was transmitted successfully at least once. If the batch was flagged multitransmittable, this flag is usually not turned on after the batch is successfully transmitted.
U	Batch is permanently unextractable. Not applicable with the FLAGS parameter (for enhanced filtering purposes). Not available with the FLAGS parameter.
X	Batch was transmitted using Bisync with **CODE=A** or **CODE=B** (transparent mode Bisync) and **INDEX=YES**.

The following table explains each protocol flag:

Protocol Flags	Description
A (TCP)	Batch originated at the host site
B (BSC)	Batch originated at a remote Bisync site
F (FTP)	Batch originated at a remote FTP site
G (SSL)	Batch was added with SSL (Secure FTP), AS2, WebDAV.
H (AS2, WebDAV)	Batch was added by the HTTP daemon. The HTTP daemon adds batches for both the AS2 and WebDAV protocols.
J (BP)	Batch was added by a GIS business process.
L (SSH SFTP)	Batch was added with SSH SFTP or SCP.
Q (ASY)	Batch originated at a remote Async site
V (BP)	Batch was acknowledged by the GIS business process.
W (AS2)	Batch was added by the EDIINT daemon.

The following table explains each data format flag. The data format flags follow transparency. If data is received transparently, the batch is flagged as binary. Therefore, the inbound transparency sets the data format flag.

Data Format Flag	Description
Y (BIN)	Binary. Not available with the FLAGS parameter.
Z (ASC)	ASCII. Not available with the FLAGS parameter.
K (EBC)	EBCDIC (nontransparent text). Not available with the FLAGS parameter.

Wildcard Syntax

The following wildcard syntax characters are supported for Async, Bisync, and FTP commands.

Syntax Character	Indicates
?	Matches any one character at the end of a string. For example, 2? selects 22 or 29 or 2a or 2 followed by any other nonzero character.
*	Matches zero or more characters. For example, 2* selects 2 or 23 or 2a or 227.
[ ]	Matches whatever is inside the brackets. This syntax can designate ranges. For example, [2*3] would select 23 and 24563 and 233 and so forth. Also [b-k,p] selects everything listed between and including b through k and p. The notation [abc]* selects everything that starts with a, b, or c.

Chapter 2

Standard FTP Commands

The Connect:Enterprise FTP server accepts commands in two formats: Mailbox ($$) syntax and standard FTP syntax.

This chapter describes the following commands available to FTP remote users using standard FTP syntax. Refer to Chapter 3, *FTP $$ Commands*, for information about FTP commands using $$syntax.

- **cd**—Changes the working mailbox ID.
- **delete** or **del**—Flags a batch of data as deleted.
- **dir**—Requests a formatted listing of batches from the host site.
- **get**—Requests a batch of data from the host site.
- **mdelete** or **mdel**–Flags multiple batches of data as deleted.
- **mget**—Requests multiple batches of data from the host site.
- **mput**—Sends multiple batches of data to the host site.
- **password**—Changes your Connect:Enterprise password.
- **put**—Sends a batch of data to the host site.
- **pwd**—Prints the working mailbox ID.
- **rename**—Changes the name of a batch.

The **user**, **binary** (**bin**), **ascii**, **prompt**, and **runique** commands are also available to users of the Connect:Enterprise FTP server. Before using any commands, ensure that you have met the prerequisites listed in the following section.

Requirements

Obtain the following information from your host site system administrator prior to executing a Connect:Enterprise FTP command:

- Mailbox ID

- Password and any additional site-specific security, including information about *mbxacl.conf*, which restricts users from performing specific commands in specific mailbox IDs other than their own.
- IP address or host name and FTP port
- Secure FTP requirements
- To use Secure FTP, both the sending and receiving sites must have FTP software that is capable of using Secure Sockets Layer (SSL) protocol.

Logging On to Connect:Enterprise

If your remote site uses FTP communications software, use the commands and procedures that follow to access and use Connect:Enterprise:

1. At the command line prompt for your system (shown as $ here), type this command and parameters:

```
$ ftp host_name port_number
```

Parameter	Description
host_name	The name of the system running Connect:Enterprise. You can enter the IP address of the host instead of the host name.
port_number	The Connect:Enterprise FTP port listener number. Refer to the Define System functionality of the Site Administration user interface to specify this port.

Because some FTP clients do not allow arguments on the UNIX FTP command line, an alternative way to access Connect:Enterprise is to enter the **FTP** command without arguments, followed by the **open** command at the ftp> prompt, as shown in the following example:

```
$ ftp
ftp> open host_name port_number
```

The host site prompts you for a login ID, as shown in the following example.

```
Name (host_name:default_userid): logon_ID
```

2. Enter a logon ID. This must be defined in an account definition by a Connect:Enterprise site administrator. If a password is required for your site, it also must be entered in the account definition.

After you have entered a logon_ID, you are prompted for a password, as shown in the following example:

```
Password:
```

The following is an example of a remote user on a workstation called *bevo* logging on to the Connect:Enterprise FTP server on a local (host) workstation called *local*, where:

- The command line prompt on the remote system is bevo>.
- The host_name argument to the **FTP** command is *local.*
- The port_number argument is 10031.
- The user name (logon_ID) is mary.
- The default_userid (steve) is different from the logon_ID (mary).

If the logon_ID matches the default_userid, pressing **Enter** is sufficient. In this example, however, the logon_ID does not match the default_userid. As you enter the password, it is not displayed. All examples show user inputs in bold.

```
bevo> ftp local 10031
Connected to local.group.company.com.
220 <<<Connect:Enterprise n.n.nn>>> at local FTP server ready.
Time = 16:02:42
Name (labs:steve): mary
331 Password required for mary.
Password:
230 Connect:Enterprise login ok, access restrictions apply.
ftp>
```

You now have access to Connect:Enterprise.

After you are connected to the host site system, you can send data to and receive data from the repository with the commands described in the following sections.

Changing Your Password

There are two messages associated with changing your password:

✦ If your password is about to expire, you will get a message similar to the following:

```
230 Connect:Enterprise UNIX login ok, access restrictions apply, password expires
in 1 day.
```

In this case, the next time you log on, type the following at the password prompt:

```
Password: oldpassword/newpassword/newpasssword
```

Where *oldpassword* is your expired password and *newpassword* is the password you want to change it to.

- If your password is expired, your log on attempt will fail with a message similar to the following:

```
539 User $$user01 login incorrect, password change required
Login failed.
```

In this case, attempt to log on again. At the password prompt, type the following:

```
Password: oldpassword/newpassword/newpasssword
```

Where *oldpassword* is your expired password and *newpassword* is the password you want to change it to.

Logging Off FTP

When you are finished using the Connect:Enterprise FTP server, you must log off using the **bye** or **quit** command. In the following example, the user enters **bye**, receives a 221 message (Goodbye), and returns to the command line prompt (bevo>).

```
ftp> bye
221 Goodbye.
bevo>
```

cd Command

The **cd** (change directory) command changes the current working mailbox ID to the one specified by the argument. If the mailbox ID does not exist, **cd** warns the user that no batches exist in the mailbox (but allows the user to stay there anyway). The **cd** command only allows users to access mailboxes specified by the **Mailbox list** parameter in their account definition.

The permissions for mailbox IDs are set by the system administrator in the *mbxacl.conf* file.

cd Command Format

You can use only the standard FTP syntax for the **cd** command, as shown in the following example:

```
ftp> cd /mailbox_ID
```

Note: Enter **cd** commands with a forward slash in front of the argument unless you are in the root directory (/).

Standard FTP cd Syntax

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
/mailbox_ID	identifies the name of the mailbox ID that becomes the working mailbox ID. This parameter can be up to eight characters. It must be preceded with a forward slash (/) unless you are in the root directory (/).

Standard FTP cd Examples

The first example shows the user changing directory to root (/). When the current working directory is / in the **Mailbox list** parameter in the account definition, the user only has access to the **dir** command.

If the keyword **Mailbox list** is not specified in the account definition or is explicitly defined as "*", you cannot change to the root directory. The message to the user is shown as follows:

```
ftp> cd /
550 CWD failed: MAILBOX_LIST is "*" (the default).
```

If the remote user's account definition contains a **Mailbox list** specification, such as "mbx1[,mbx2...]", then the remote user can enter **cd /** and enter the **dir** command (only). The **dir** command shows only mailboxes that appear in **Mailbox list** for this remote user (in the account definition).

The following example shows a user named steve, changing directory to root and obtaining a directory listing; **Mailbox list** is defined in user steve's account definition as "steve1,other_id", so the only mailboxes steve can see are steve1, other_id, and his own login ID (steve).

```
ftp> cd /
250 CWD to '/' successful
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for .
d   -   -   -  -  -  -  - steve1
d   -   -   -  -  -  -  - other_id
d   -   -   -  -  -  -  - steve

Total number of Mailboxes = 3
226 Transfer complete.
ftp>
```

If **Mailbox list** is defined as "", the user is allowed to change to the root directory (/) with **cd**, but the only mailbox listed in the directory is the user's login mailbox (equivalent to user's name). This is shown in the following example for user steve:

```
ftp> cd /
250 CWD to '/' successful
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for .
d   -   -   -  -  -  -  - steve

Total number of Mailboxes = 1
226 Transfer complete.
ftp>
```

The following example shows a remote user changing directory to a mailbox ID called his_id. If his_id is not in **Mailbox list** for this remote user in the account definition, the system displays the warning message *Permission denied*.

```
ftp> cd /his_id
550 /his_id: Permission denied.
ftp>
```

This example shows a remote user changing directory to a nonexistent mailbox ID called no_exist.

```
ftp> cd /no_exist
250 CWD successful.
ftp>
```

The following message is displayed:

```
250 CWD successful.
```

You can then use the **put** command to place a batch into the new mailbox ID. This command only works if no_exist is listed in the user's **Mailbox list**.

delete Command

The **delete** command flags a batch for deletion from the mailbox, including unrequestable batches.

The permissions for your mailbox ID are set by the system administrator in the *mbxacl.conf* file. These permissions determine the mailbox IDs from which you can delete batches.

The **delete** command can be abbreviated to **del**. After Connect:Enterprise receives the **del** command, it processes the command and returns the results to the remote site. The remote site collects the output to the console.

When a batch is flagged for deletion, the D flag is set for that batch. The host site operator can restore the batch (using the **cmustatus** command) or physically erase it (using the **cmuerase** command).

del Command Format

The following example shows the standard FTP syntax:

```
ftp> del[ete] [/mailbox_ID/]batch_ID
```

Note: The **delete** command can have an unexpected result under certain conditions. Refer to the *Considerations When Using the del, mdel, get, and mget Commands* on page 39 for more information on how to avoid this potential problem.

Standard FTP delete Syntax

The **del** command has one required parameter, the batch_ID. If you enter the **del** command without a batch ID argument, you are prompted for a remote file.

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
batch_ID	identifies the user batch ID or the batch number. The user batch ID can be 1–64 characters. If the batch ID contains spaces, it must be enclosed in double quotation marks. If the batch ID has no spaces, quotation marks are not needed. Wildcard specifications (like an asterisk, *) are allowed. To specify a batch number, use the pound sign (#), followed by the batch number, for example, #29.
mailbox_ID	indicates a mailbox ID. It precedes the batch ID. If the mailbox_ID differs from the login ID (remote ID) and is not specified in the **Mailbox list** in the remote user's account definition, the **del** command returns the error message *Permission denied.* The default is the current working directory (mailbox ID). Refer to the Define Accounts function in the Site Administration user interface Help for a complete description of the **Mailbox list** parameter.

Standard FTP delete Examples

The next example shows how to logically delete the batch in the current working mailbox with the batch ID *old batch*. Note the double quotation marks are required because the batch ID contains a space.

```
ftp> delete "old batch"
250 DELE command successful.
```

In the following example, the user deletes all batches with the batch ID *his_batch* and the mailbox ID *his_id*. To delete a single batch from a list, use **mdelete** with FTP interactive mode enabled (using the **prompt** command). The **del** command does not prompt; it deletes all batches with the specified batch ID.

```
ftp> del /his_id/his_batch
```

dir Command

The **dir** (directory) command displays information about batches in the current working mailbox.

Note: The **ls** command can be used in place of **dir** but will return a short listing of the batch ID only. The **ls** command does not display batches with the deleted (D), transmitted (T), or not requestable (!R) flag set.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file. These permissions determine the mailboxes for which you can obtain directory listings. If you enter a **dir** command for a mailbox to which you do not have access, you get the message *Permission denied.*

The **dir** command does not display batches with the delete (D) flag set. The directory listings are either displayed to standard output, or written to a local file if one is specified as an argument.

dir Command Format

The following example shows the standard FTP syntax:

```
ftp> dir [[/mailbox_ID/]batch_ID] [local_filename]
```

The following table describes the parameters, required parameters are in bold.

Parameter	Description
batch_ID	identifies the user batch ID or the batch number. The user batch ID can be 1–64 characters. If the batch ID contains spaces, it must be enclosed in double quotation marks. If the batch ID has no spaces, quotation marks are not needed. Wildcard specifications (like an asterisk, *) are supported. To specify a batch number, use the pound sign (#), followed by the batch number, for example, #29.
mailbox_ID	identifies the 1–8 character mailbox ID for which directory information is desired. Connect:Enterprise searches for all batches that match the specified mailbox ID. Type the ID surrounded by forward slashes (indicating a path). Wildcard specifications (like an asterisk, *) are supported. When you specify mailbox ID, you can use double quotation marks in two ways. They can enclose the batch ID (following the mailbox ID), or they can enclose the mailbox ID and the batch ID. The two methods are shown in the following example: ftp> dir /other/“some batch” ftp> dir “/other/some batch” Both commands list all batches with batch ID of *some batch* in the /other mailbox. If the mailbox ID is not the login ID (remote ID) and the account definition for the remote does not have the specified mailbox in its **Mailbox list** specification, **dir** produces the error message: /*mailbox_ID*: *Permission denied.* The default is the current working directory (mailbox ID).

Parameter	Description
local_filename	identifies the local destination file name (on the remote user's host) where the directory listing of batches is written. Include path names when necessary. When a file name is not specified, the batch information is returned to the remote site's display. If no file name is used, output can still be collected to a file by conducting the entire FTP session from a script file with output redirected to a specified file.

Output

After a **dir** command is issued, Connect:Enterprise checks the repository for the designated batches and returns that information to the remote site in the following format:

```
flags protocol batch_type originator batch_no size time batch_id
```

The output includes the following parameters:

Parameter	Description
Flags	The 10-character string where each character indicates a unique characteristic about the batch. Each flag has a specific place in the string. If the flag does not apply to (is set for) this batch, a dash (-) is displayed instead of a letter. Process flags are always displayed in the same order on the report. For example, the first 10 places can read: IA----UN--. I (Incomplete) or - A (batch was added with **cmuadd** by the local site) or C (collected from the remote site) R (Requestable) or - T (Transmitted) or - E (Extracted) or - M (Multitransmittable) or - U (Unextractable) or - N (Nontransmittable) or - P (in Progress) or - **Note**: The tenth character is always a dash to separate the flags from the protocol. Chapter 1, *Accessing Connect:Enterprise*, contains definitions of the flags.
protocol	indicates the transmission protocol used to place the batch into the specified mailbox. The following options are possible: TCP Transmitted locally with **cmuadd** command ASY Transmitted remotely with Async protocol BSC Transmitted remotely with Bisync protocol FTP Transmitted remotely with FTP protocol FTS Transmitted remotely with Secure FTP protocol --- Activity log batch

Parameter	Description
batch_type	A single character, which can be one of three values: A (ASCII), B (binary), or E (EBCDIC).
originator	The originator ID for the batch.
batch_no	The batch number.
size	The size of the batch in bytes.
time	The time the batch was created.
bid	The batch ID.

Standard FTP dir Examples

In the following example, the user does not specify any arguments. Because no mailbox or batch IDs are provided, the **dir** command displays the contents of the current working mailbox, which consists of a batch called *test.c*, and the activity log batch.

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for .
-C--E-----FTP A steve      2      41 Sep 02 13:47 test.c
-SR--M------- A steve      1     369 Sep 02 13:47 <<ACTIVITY LOG>>
Total number of batches listed: 2
226 Transfer complete.
ftp>
```

The following example shows the user entering the **dir** command in a mailbox that currently contains no batches.

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for .
Total number of batches listed: 0
226 Transfer complete - no batches
ftp>
```

The next example specifies batches with the batch ID *new year resolu* in the /group mailbox. If the appropriate permissions for the /group mailbox ID are set in *mbxacl.conf*, the list of batches is written to a local file called my_out.file. If my_out.file is not specified, the listing is displayed to standard output.

```
ftp> dir /group/"new year resolu" my_out.file
```

The following example lists all batches whose batch ID begins with the letter *b*. The listing goes to a local file called my_out.file.

```
ftp> dir b* my_out.file
output to local-file: my_out.file? y
200 PORT command successful.
150 Opening ASCII mode data connection for b*.
226 Transfer complete.
ftp>
```

Note: Using a wildcard search, such as **dir /*/***, for mailbox IDs, is not recommended. The output from the standard FTP **dir** command does not indicate the mailbox ID, so you cannot tell which batches are in which mailbox ID.

The following example displays all batches in the mailbox ID called his_mbx. For this remote user to obtain such a listing, the system administrator must first set the appropriate permissions for the his_mbx mailbox ID in the *mbxacl.conf* file. If the user does not have access to his_mbx, the message *Permission denied* is returned, which is shown in the next to the last example with the /othermbx mailbox ID below.

```
ftp> dir /his_mbx
```

The next example displays all batches in the mailbox ID called his_mbx with a batch ID that starts with the letter *x*. The same restrictions apply here as in the preceding example.

```
ftp> dir /his_mbx/x*
```

The next example illustrates failing to obtain a directory listing of a mailbox to which the user does not have access because the system administrator set parameters in the *mbxacl.conf* file in the Configurator in one of the following ways:

- Defined othermbx without the **Directory** parameter set to the **R** or **F** permissions, restricting directory access by other users
- Defined DEF_ACL (the default access control list keyword), set the **Directory** parameter to the **R** or **F** permissions, and not specified othermbx at all, in which case othermbx inherits the definition of DEF_ACL.

```
ftp> dir /othermbx
200 PORT command successful.
550 othermbx: Permission denied.
ftp>
```

The next example lists the batches in the current working mailbox (because a mailbox ID was not specified) whose batch ID is *receipt*.

```
ftp> dir receipt
```

get or recv Command

The **get** or **recv** command requests a single file from the repository. This single file contains all the batches matching the selection criteria.

The **get** or **recv** command copies one or more batches from the repository into a file in the local directory. Multiple batches with the same batch ID are always concatenated into a single file, no matter what the keyword **Outbound batch separation** is set to in the account definition for the remote site. To get the batches individually, you must use the **mget** command with FTP interactive mode enabled.

Your initial mailbox ID is determined by your FTP *logon ID* but can be changed with the **cd** command. Refer to the **cd** command on *cd Command* on page 18.

The permissions for your mailbox ID are set by the system administrator in the *mbxacl.conf* file. This file indicates mailboxes from which you can retrieve batches.

If the data format is in EBCDIC or binary format, the remote user must enter the FTP command **bin** before issuing the **get** command. For ASCII data format, enter the **asc** FTP command.

get or recv Command Format

The following example shows the standard FTP syntax for the **get** or **recv** command:

```
ftp> get|recv [/mailbox_ID/]batch_ID [dest_filename]
```

Note: The **get** command can have an unexpected result under certain conditions. Refer to the *Considerations When Using the del, mdel, get, and mget Commands* on page 39 for more information on how to avoid this potential problem.

Standard FTP get Syntax

The following table describes the parameters available. Required parameters are in bold.

Parameter	Description
batch_ID	identifies the user batch ID or the batch number. The user batch ID can be 1–64 characters. If the batch ID contains spaces, it must be enclosed in double quotation marks. If the batch ID has no spaces, quotation marks are not needed. Wildcard specifications (like an asterisk, *) are supported. To specify a batch number, use the pound sign (#), followed by the batch number, for example, #29.

Parameter	Description
dest_filename	indicates the name of the local file used to store the batch. If not specified, the default is the batch ID specified in the first argument (batch_ID). Depending on the file system being used and the use of the **Remote filename** parameter, the **get** command can truncate file names (for example, to the 8.3 format on DOS). Therefore, always supply the dest_filename argument. Refer to the Define Accounts function in the Site Administration user interface Help for a complete description of the **Remote filename** parameter. It precedes the batch ID specification. Type the mailbox ID surrounded by forward slashes (indicating a path).
mailbox_ID	is a mailbox ID. If the mailbox ID is not the login ID (remote ID) and is not specified in **Mailbox list** in the remote user's account definition, then the **get** command returns the error message *Permission denied.* The default is the current working directory (mailbox ID).

Standard FTP get Examples

In the following example, the optional *dest_filename* argument is not specified, so a batch called *new_bid* is extracted from the current working mailbox ID into a local file also called *new_bid*.

```
ftp> get new_bid
200 PORT command successful.
150 Opening ASCII mode data connection for new_bid.
226 Transfer complete.
45 bytes received in 0.3938 seconds (0.1116 Kbytes/s)
ftp>
```

In the next example, the *dest_filename* argument is specified, so a batch whose batch ID is *monthly_rpt* is extracted from the current working mailbox ID to a file named *rept.doc*. If multiple batches have this batch ID, they are concatenated into a local file named *rept.doc* (to extract them individually, use the **mget** command with interactive mode enabled).

```
ftp> get monthly_rpt rept.doc
200 PORT command successful.
150 Opening ASCII mode data connection for monthly_rpt.
226 Transfer complete.
45 bytes received in 0.3838 seconds (0.1145 Kbytes/s)
ftp>
```

In the next example, o*ther_id* is some mailbox ID other than the login ID, and /other_id exists as a directory on the local system. The batches with the batch ID *new.rpt* are extracted from the other_id mailbox and concatenated to a local file also named *new.rpt* in the /other_id directory on the local system (because the *dest_filename* argument is not specified). This is true only if the other_id mailbox ID is set appropriately in the *mbxacl.conf* file or if the mailbox daemon, **cmumboxd**, was brought up without the **-r** option.

```
ftp> get /other_id/new.rpt
```

The next example is similar to the preceding example, with one exception: /other_id does not exist as a directory on the local system. The **get** command returns an error.

```
ftp> get /other_id/new.rpt
/other_id/new.rpt: No such file or directory
ftp>
```

Specify a local file name as a second argument to the **get** command, even if it is the same base name as the batch ID (*new.rpt*), as shown in the following example. The file is put in the user's local working directory (where FTP started or the last **lcd** command was issued).

```
ftp> get /other_id/new.rpt new.rpt
200 PORT command successful.
150 Opening ASCII mode data connection for /other_id/new.rpt.
226 Transfer complete.
45 bytes received in 0.4212 seconds (0.1043 Kbytes/s)
ftp>
```

In this example, the retrieved batches have the batch ID *receipt*.

```
ftp> get receipt
```

Unavailable Batches

Batches with the following characteristics are unavailable to remote users. The **get** command does not retrieve batches that are:

- Flagged as nontransmittable (U)
- Flagged as deleted (D)
- Not flagged as requestable (R)
- Flagged as transmitted (T)
- Flagged as incomplete (I)
- Not matching the mailbox ID and/or user batch ID

mdelete Command

The **mdelete** command logically deletes multiple batches from the repository (including unrequestable batches) by setting the D flag on these batches. This command can be abbreviated to **mdel**.

The permissions for your mailbox ID are set by the system administrator in the *mbxacl.conf* file. This file indicates the mailbox IDs from which you can delete batches.

The main advantage of using the **mdel** command instead of the **del** command is that with interactive mode enabled, the **mdel** command prompts you prior to deleting each batch, whereas the **del** command deletes all batches without prompting. If you do not want prompts, use the **del** command.

mdel Command Format

You can only use standard FTP syntax with the **mdel** command.

```
ftp> mdelete [/mailbox_ID/]batch_ID
```

Note: The **mdelete** command can have an unexpected result under certain conditions. Refer to the *Considerations When Using the del, mdel, get, and mget Commands* on page 39 for more information on how to avoid this potential problem.

The following table describes the parameters. Required parameters are in bold.

Parameter	Description
batch_ID	identifies the user batch ID or the batch number. The user batch ID can be 1–64 characters. If the batch ID contains spaces, it must be enclosed in double quotation marks. If the batch ID has no spaces, quotation marks are not needed. Wildcard specifications (like an asterisk, *) are supported. To specify a batch number, use the pound sign (#), followed by the batch number, for example, #29.
mailbox_ID	is a mailbox ID. It precedes the batch ID specification. If the mailbox ID is not the login ID (remote ID) and is not specified in the **Mailbox list** specification in the user's account definition, then the **mdelete** command returns the error message *Permission denied* The default is the current working mailbox ID.

Standard FTP mdel Examples

The **mdelete** command in the following example sets the D flag for all batches in the current working mailbox with the batch ID beginning with *old_batch*.

```
ftp> mdelete old_batch*
```

In the following example, the **mdelete** command sets the D flag for all batches with batch ID *some.text* in the mailbox with mailbox ID *his_id*.

The **mdelete** command succeeds only if one of the three following conditions is met:

- **cmumboxd** is started without the **-r** option (indicating that all remote users have full access to all batches in all mailbox IDs).
- The his_id mailbox is specified in the *mbxacl.conf* file with the **Delete** parameter set to the **R** or **F** permissions.

- The parameter his_id is not specified in the *mbxacl.conf* file, but DEF_ACL has the **Delete** parameter set to the **R** or **F** permissions.

```
ftp> mdelete /his_id/some.text
```

In the following example, the current mailbox has three batches with the batch ID *new*. Although they all have the same batch ID, each batch has a unique batch number. When interactive mode is enabled, the **mdelete** command prompts for confirmation of the delete operation for each batch, whereas the **del** command deletes all batches with the specified batch ID.

```
ftp> mdel new
```

The following example shows how to delete batches that have batch IDs containing spaces. Note the use of double quotation marks.

```
ftp> mdel "batches with spaces"
```

The next example shows the actual interaction between the user and Connect:Enterprise. Bold font indicates the user's input. First, the user enters a **dir** command, which lists three batches (plus the activity log). This information is provided for clarity.

Next, the user enters the **mdel *** command. Because interactive mode is enabled (it can be toggled with the **prompt** command), the **mdel *** command prompts to delete each batch.

Note: The activity log is not deleted.

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for .
-C--E-----FTP A steve       15     1193 Aug 29 07:52 eob.sh
-C--E-----FTP A steve       27     3039 Aug 29 07:52 a.out
-C--E-----FTP A steve       24       60 Aug 29 08:31 bscrmt
-SR--M------- A steve        1     1028 Aug 29 08:31 <<ACTIVITY LOG>>
Total number of batches listed: 4
226 Transfer complete.
ftp> mdel *
mdel eob.sh? y
250 DELE command successful.
mdel a.out? y
250 DELE command successful.
mdel bscrmt? y
250 DELE command successful.
ftp>
```

mget Command

The **mget** command results in a file being created on the local system corresponding to each batch in the repository that matches the selection criteria.

When used with standard FTP syntax, the **mget** command copies multiple batches from a mailbox to the current working directory on the remote user's local node. The arguments are one or more batch IDs, each optionally preceded by a mailbox ID.

The permissions for your mailbox ID are set by the system administrator in the *mbxacl.conf* file. These permissions determine the mailbox IDs from which you can retrieve batches.

Your initial mailbox ID is determined by your FTP *logon ID* but can be changed with the **cd** command. Refer to the **cd** command on *cd Command* on page 18. If the data format is EBCDIC or binary, the remote user must enter the FTP command **bin** before issuing the **mget** command. For ASCII data format, enter the **asc** FTP command, which is the default.

mget Command Format

The following shows the standard FTP syntax for the **mget** command:

```
ftp> mget [/mailbox_ID/]batch_ID...
```

Note: The **mget** command can have an unexpected result under certain conditions. Refer to the *Considerations When Using the del, mdel, get, and mget Commands* on page 39 for more information on how to avoid this potential problem.

Standard FTP mget Syntax

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
batch_ID	identifies the user batch ID or the batch number. The user batch ID can be 1–64 characters. If the batch ID contains spaces, it must be enclosed in double quotation marks. If the batch ID has no spaces, quotation marks are not needed. Wildcard specifications (like an asterisk, *) are supported. To specify a batch number, use the pound sign (#), followed by the batch number, for example, #29. Each batch ID argument to the **mget** command can be preceded by a mailbox ID surrounded by forward slashes.

Parameter	Description
mailbox_ID	is a mailbox ID. It precedes the batch ID specification. Type the mailbox ID surrounded by forward slashes (indicating a path). If the mailbox ID is not the login ID (remote ID) and is not specified in the **Mailbox list** parameter in the user's account definition, and **cmumboxd** is started with the **-r** option, then the **mget** command returns the error message *Permission denied*. If this parameter is not specified, the default is the current working directory (mailbox ID).

If the **Outbound batch separation** parameter is set either to NO or OPT3 in the account definition, the **mget** command extracts batches to files with file names that match the batch ID. If the **Outbound batch separation** parameter is set to **Batches are created...** and multiple batches are specified (for example, bid1 bid2 bid3...), the **mget** command extracts each batch to a file named *batch_id.batch_number* on UNIX systems or *batch_number*.DAT on DOS systems. Refer to the Define Accounts function in the Site Administration user interface Help for a complete description of the **Outbound batch separation** parameter.

Standard FTP mget Examples

In the following example, one batch ID is specified: his_batch. A mailbox ID (/other_id/) is specified. If other_id does not match the user's login ID, the **mget** command succeeds only if the operator has access to the other mailbox ID; that is, the other_ID mailbox is set appropriately in *$CMUHOME/med/mbxacl.conf* and other_id is included in the **Mailbox list** specification in the account definition for this remote user. If so, the batch his_batch is retrieved. Otherwise the **mget** command returns the error message *Permission denied*. In this example, interactive mode is enabled by default, so the FTP prompts the user before receiving the batch.

```
ftp> mget /other_id/his_batch
mget /other_id/his_batch? y
200 PORT command successful.
150 Opening ASCII mode data connection for /other_id/his_batch.
226 Transfer complete.
45 bytes received in 0.3673 seconds (0.1196 Kbytes/s)
ftp>
```

Note: Unlike the **get** command examples shown earlier, the preceding example works (on AIX) even if the directory /other_id does not exist on the remote user's system. However, on HP-UX systems, the **mget** command behaves like the **get** command: if the directory /other_id does not exist on the local system, the preceding **mget** command fails. Instead, use the **cd** command to change to /other_id mailbox, then issue the **mget** command without the mailbox ID.

In the following example, the **Outbound batch separation** parameter is set to either **No separation...** or **Multiple batches are sent...**. A mailbox ID is not specified. This example extracts batches with batch IDs of a.c, x.c, and y.c from the current mailbox to three separate files named a.c, x.c, and y.c on the remote user's local system. Remote users are prompted before each individual file is retrieved (use the **prompt** command to disable prompting).

```
ftp> mget a.c x.c y.c
mget a.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for a.c.
226 Transfer complete.
45 bytes received in 0.3594 seconds (0.1223 Kbytes/s)
mget x.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for x.c. 226 Transfer complete.
45 bytes received in 0.3908 seconds (0.1124 Kbytes/s)
mget y.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for y.c. 226 Transfer complete.
45 bytes received in 0.4014 seconds (0.1095 Kbytes/s)
ftp>
```

In the following example, the **Outbound batch separation** parameter is set to either **No separation...** or **Multiple batches are sent...** A mailbox ID is not specified. In this instance, the **mget** command extracts a batch called *new report* to a file called *new report*. It extracts all batches with batch IDs beginning with the pattern *labs* (for example, *labs_resume*, *labs 1997 plan*, *labs.xyz*, and so forth). Finally, it extracts all batches with batch IDs beginning with the letter *g*.

```
ftp> mget "new report" labs* g*
```

In the following example, the **Outbound batch separation** parameter is set to **Batches are created...** in the user's account definition. Three batches have the batch ID *labs_97* and their batch numbers are 3, 3001, and 99. No mailbox ID is specified.

If the **File format on remote site** parameter is set to **Free format UNIX style...** in the account definition, these three batches are extracted to files named labs_97.3, labs_97.3001, and labs_97.99, respectively. If the **File format on remote site** parameter is set to **8.3 DOS style...** in the user's account definition, these three batches are extracted to files named 00000003.dat, 00003001.dat, and 00000099.dat, respectively.

```
ftp> mget labs_97
```

In the following example, the **Outbound batch separation** parameter is set to **Batches are created...** in the user's account definition. Mailbox IDs are not specified in front of any of the batches. Two batches have batch ID *smith_Lee*, with batch numbers of 11 and 9023, and three batches have batch ID *smith_davis,* with batch numbers of 400, 2134, and 1009876. If the **File format on remote site** parameter is set to **Free format UNIX style...** in the account definition, these five batches are extracted to files named smith_Lee.11, smith_Lee.9023, smith_davis.400, smith_davis.2134, and smith_davis.1009876, respectively. If the **File format on remote site** parameter is set to **8.3 DOS style...** in the account definition, these batches are extracted to files named 00000011.dat, 00009023.dat, 00000400.dat, 00002134.dat, and 10009876.dat, respectively. This process is repeated for the various batches with batch IDs beginning with the letter *g*.

```
ftp> mget smith* g*
```

mput Command

The **mput** command enables users to copy multiple files from the local system to the mailbox. It creates multiple batches whose batch IDs match the base name of the local files, and places these batches into the current working mailbox ID. If the data format is EBCDIC or binary, the remote user must enter the FTP command **bin** before issuing the **mput** command. For ASCII data format, enter the **asc** FTP command.

The permissions associated with all mailbox IDs are set by the system administrator in the *mbxacl.conf* file in the *$CMUHOME/med* directory. These permissions determine the mailbox IDs to which you can add batches. Your initial mailbox ID is determined by your FTP *logon ID* but can be changed with the **cd** command. Refer to the **cd** command on *cd Command* on page 18.

mput Command Format

Only one syntax can be used for the **mput** command. The following is the standard FTP syntax for the **mput** command, with ftp> as the prompt:

```
ftp> mput source_path
```

Standard FTP mput Syntax

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
Rsource_path	indicates the path name for the source files you want added to Connect:Enterprise. This can include wildcards, as shown in the following examples.

Standard FTP mput Examples

The following example creates a batch for each file in the local directory and places all these batches into the current working mailbox ID. This example uses four local files (a.c, test.c, x.c, and y.c); consequently, four new batches are created. Interactive mode is enabled by default.

```
ftp> mput *
mput a.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for a.c.
226 Transfer complete (Batch Number = 18).
45 bytes sent in 0.001574 seconds (27.92 Kbytes/s)
mput test.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for test.c.
226 Transfer complete (Batch Number = 19).
45 bytes sent in 0.000704 seconds (62.42 Kbytes/s)
mput x.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for x.c.
226 Transfer complete (Batch Number = 20).
45 bytes sent in 0.000696 seconds (63.14 Kbytes/s)
mput y.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for y.c.
226 Transfer complete (Batch Number = 21).
45 bytes sent in 0.000699 seconds (62.87 Kbytes/s)
ftp>
```

In the next example, x.doc, y.doc, and z.doc exist in */users/smith*. Then you have created three batches whose batch IDs are x.doc, y.doc, and z.doc, and placed these three batches in the current working mailbox ID.

```
ftp> mput /users/smith/*.doc
```

Interactive mode is enabled and disabled with the **prompt** command and is enabled by default. When interactive mode is enabled, the **mput** command prompts you before making a new batch for each local file. If interactive mode is disabled, the **mput *** command creates a batch for each of the files.

The next example shows the interaction between the user and the Connect:Enterprise FTP server with interactive mode enabled; the user's input is in bold.

```
ftp> mput *
mput abc.c? y
200 PORT command successful.
150 Opening ASCII mode data connection for abc.c.
226 Transfer complete (Batch Number = 9).
8576 bytes sent in 0.05533 seconds (151.4 Kbytes/s)
mput def.sh? y
200 PORT command successful.
150 Opening ASCII mode data connection for def.sh.
226 Transfer complete (Batch Number = 15).
1228 bytes sent in 0.01828 seconds (65.62 Kbytes/s)
mput a.out? y
200 PORT command successful.
150 Opening ASCII mode data connection for a.out.
226 Transfer complete (Batch Number = 27).
3047 bytes sent in 0.01803 seconds (165 Kbytes/s)
mput doc.txt? n
mput exits.diff? n
mput FTP.trace? n
mput test.c? n
ftp>
```

put Command

The **put** command adds a single batch of data to a mailbox ID. If the data format is EBCDIC or binary, the remote user must enter the FTP command **bin** before issuing the **put** command. For ASCII data format, enter the **asc** FTP command. These options are similar to the FTP **put** parameters for FTP $$ commands shown on *put Command* on page 73. Refer to the Define Accounts or Define Schedule function in the Site Administration user interface for a complete description of the **FTP put options**.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file in the *$CMUHOME/med* directory. The contents of this file determine the mailboxes you can add batches to.

put Command Format

The following is the standard FTP syntax:

```
ftp> put source_path [batch_ID]
```

Standard FTP put Syntax

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
source_path	indicates the source file path for the data you want added to Connect:Enterprise.

Parameter	Description
batch_ID	identifies the 1–64 byte user batch ID for the added batch. If the batch ID is not specified, the base name (file name without the path) of the source_path is used. If the batch ID contains spaces, it must be enclosed in double quotes.

Standard FTP put Examples

The **put** command does not allow users to place a file into another mailbox ID other than the current working mailbox ID (directory). Your initial mailbox ID is determined by your FTP *logon ID* but can be changed with the **cd** command. Refer to the **cd** command on *cd Command* on page 18. To add files to other mailbox IDs, you must first change to the other mailbox ID using the **cd** command and then execute the **put** command. Parameters (source_path and batch_ID) must be separated by one or more spaces.

For example, suppose user mary has logged in to Connect:Enterprise as shown earlier. Initially, her working mailbox is /mary. Now suppose she enters the following command:

```
ftp> put source_path my_batch
200 PORT command successful.
150 Opening ASCII mode data connection for my_batch.
226 Transfer complete (Batch Number = 15).
45 bytes sent in 0.02522 seconds (1.742 Kbytes/s)
ftp>
```

This command adds the file specified by *source_path* as batch ID *my_batch* in the current working mailbox ID, which is /mary.

The following example shows how mary can add a file from a different local directory (*/users/smith*) to her mailbox ID of /mary. She adds the batch with batch ID *text.file* to the current working mailbox ID, /mary. The batch ID (second argument) is not specified, so the base name of the file (*text.file*) is used as the batch ID.

```
ftp> put /users/smith/text.file
200 PORT command successful.
150 Opening ASCII mode data connection for /users/smith/text.file.
226 Transfer complete (Batch Number = 16).
45 bytes sent in 0.001042 seconds (42.17 Kbytes/s)
ftp>
```

The next example adds a batch to the current working mailbox ID. The batch ID is specified as my_bid. The batch contains the file specified by *text.file*.

```
ftp> put text.file my_bid
200 PORT command successful.
150 Opening ASCII mode data connection for my_bid.
226 Transfer complete (Batch Number = 17).
45 bytes sent in 0.002082 seconds (21.11 Kbytes/s)
ftp>
```

In the following example, the user has specified /his_id/ as part of the batch ID specification; however, the **put** command ignores this, and adds the file called *my.file* to the batch with the batch ID *my_bid* in the current working mailbox ID.

```
ftp> put my.file /his_id/my_bid
```

To place *my.file* in /his_id, issue the following command:

```
ftp> cd /his_id
ftp> put my.file my_bid
```

The preceding **put** command succeeds only if the *mbxacl.conf* file is configured to allow users to add batches to the /his_id mailbox ID.

pwd Command

The **pwd** command displays the current working directory (mailbox ID) at any time during an FTP session. It prints to standard output, which can be the terminal, unless scripting is being run with standard output set to a file.

pwd Command Format

You can only use standard FTP syntax with the **pwd** command.

Example

This example shows the only way the **pwd** command can be used. The command **pwd** accepts no parameters or arguments.

```
ftp> pwd
257 "/steve" is current directory.
ftp>
```

rename Command

This command renames a remote batch ID.

rename Command Format

The following example shows the FTP syntax for the **rename** command:

```
ftp> rename [/mailboxID/]oldbatchID [/mailboxID/]newbatchID
```

Standard FTP put Syntax

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
oldbatchID	Identifies the batch ID to change.
newbatchID	The new batch ID.
mailboxID	Identifies the mailbox location of the batch.

Standard FTP rename Examples

The following example renames a batch in the user01 mailbox from batch01 to 01batch:

```
ftp> rename /user01/batch01 /user01/01batch
```

The following example renames a batch in the current working mailbox from batch01 to 01batch:

```
ftp> rename batch01 01batch
```

Considerations When Using the del, mdel, get, and mget Commands

An unexpected result can occur when you use the Connect:Enterprise FTP server. It occurs only when all of the following six conditions are true:

1. The **Outbound batch separation** keyword is set to **Batches are created...** in the account definition for this remote user. OPT4 indicates that individual batches with identical batch IDs are retrieved into separate files rather than concatenated. Refer to the Define Accounts function in the Site Administration user interface Help for a complete description of the **Outbound batch separation** keyword.
2. FTP interactive mode is enabled (not disabled with the **prompt** command). You are prompted for each file when you issue an **mget** or **mdel** command.
3. The remote user issues an **mget** or **mdel** command, immediately followed by a **get** or **delete** command. Four combinations are possible: **mget** followed by **get**; **mget** followed by **delete**; **mdel** followed by **get**; and **mdel** followed by **delete**.
4. In the **get** or **delete** command (the second of the two commands the remote user issues, the user specifies a batch ID that ends with a period followed by a number (for example, ggg.11). Note that in this case, 11 is not the batch number; ggg.11 is the batch ID, and the batch number can be something different.

5. One of the batches processed in the **mget** or **mdel** command (entered previously) has a batch ID and batch number that, when combined (and separated by a period), match the batch ID specified in the **get** or **delete** command (for example, batch ID ggg and batch number 11)
6. When prompted for each file in the **mget** or **mdel** command, the user enters *n* (no) when prompted for the batch with the conflicting batch ID and enters *n* when prompted for all batches after it. A conflicting batch ID is one that ends with a period followed by the number of the batch specified in the **get** or **del** command.

The following example demonstrates this set of conditions. The system administrator-defined **Outbound batch separation** keyword is set to **Batches are created...** for this remote user (condition 1). When the **Outbound batch separation** parameter is set to **No separation...** or **Multiple batches are sent...**, for example, the **mget** command concatenates the two ggg batches into one file, which nullifies condition 1.

First, the user issues a **dir** command, and there are four batches. Two of them (batch numbers 22 and 83) have the same batch ID of ggg. Batch number 31 has batch ID hhh, and batch number 8 has batch ID uuu. Note that there is no batch in this directory whose batch ID is ggg.22.

The example *ftp> dir* on page 41 illustrates issuing an **mget** command immediately followed by a **get** command (condition 3). The prompt for each file ('mget *batch*?') indicates that interactive mode is enabled (condition 2). In the **get** command, the user attempts to get a batch whose batch ID ends with a period followed by a number (condition 4). In this case, the batch ID is ggg.22. One of the batches processed in the previous **mget** command had batch ID ggg and batch number 22, which, when combined with a period, produce ggg.22, which matches the batch ID specified in the **get** command (condition 5). In the **mget** command, when prompted for ggg.22, the user entered *n*, and for all batches after it, namely uuu.8 and ggg.83 (condition 6).

Because all the conditions are met, the **get** command retrieves the batch with the batch ID ggg and the batch number 22 rather than return the error message that the specified batch, with batch ID of ggg.22, cannot be retrieved (because it does not exist).

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for .
-AR--M----TCP A george         31     1793 Sep 08 17:22 hhh
-CR--M----FTP A george         22     1793 Sep 09 18:08 ggg
-CR--M----FTP A george          8     1723 Sep 09 18:09 uuu
-CR--M----FTP A george         83     2527 Sep 09 18:09 ggg
Total number of batches listed: 4
226 Transfer complete.
ftp> mget *
mget hhh.31? n
mget ggg.22? n
mget uuu.8? n
mget ggg.83? n
ftp> get ggg.22
200 PORT command successful.
150 Opening ASCII mode data connection for ggg.22.
226 Transfer complete.
1841 bytes received in 1.402 seconds (1.282 Kbytes/s)
ftp>
```

To avoid the problem illustrated in the preceding example, enter any command other than **mget**, **mdelete**, **get**, or **delete** after the **mget** command to nullify condition 3, or avoid using batch IDs that end with a period followed by a number.

Another way to eliminate the problem is to disable interactive mode with the **prompt** command. Note that in the following example, you are not prompted for each batch in the **mget** command, which nullifies condition 2. Therefore, when you issue the **get** command, it fails as expected, with the error message *No batches for transmission*, (because batch ID ggg.22 does not exist in this directory).

```
ftp> prompt
Interactive mode off.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for .
-AR--M----TCP A george         31     1793 Sep 08 17:22 hhh
-CR--M----FTP A george         22     1793 Sep 09 18:08 ggg
-CR--M----FTP A george          8     1723 Sep 09 18:09 uuu
-CR--M----FTP A george         83     2527 Sep 09 18:09 ggg
Total number of batches listed: 4
226 Transfer complete.
ftp> mget *
200 PORT command successful.
150 Opening ASCII mode data connection for hhh.31.
226 Transfer complete.
1841 bytes received in 1.406 seconds (1.279 Kbytes/s)
200 PORT command successful.
150 Opening ASCII mode data connection for ggg.22.
226 Transfer complete.
1841 bytes received in 1.388 seconds (1.295 Kbytes/s)
200 PORT command successful.
150 Opening ASCII mode data connection for uuu.8.
226 Transfer complete.
1779 bytes received in 1.346 seconds (1.291 Kbytes/s)
200 PORT command successful.
150 Opening ASCII mode data connection for ggg.83.
226 Transfer complete.
2624 bytes received in 1.84 seconds (1.393 Kbytes/s)
ftp> get ggg.22
200 PORT command successful.
550 *** ERROR *** NO BATCHES FOR TRANSMISSION
ftp>
```

In the following example, interactive mode is re-enabled (so condition 2 applies). In the same directory (four batches) as before, you nullify condition 6 by entering *y* for a batch (batch ID uuu, batch number 8) that follows the conflicting batch (batch ID ggg, batch number 22) in the **mget** prompting order. When you attempt to obtain batch ID ggg.22, the **get** command fails.

```
ftp> prompt
Interactive mode on.
ftp> mget *
mget hhh.31? n
mget ggg.22? n
mget ggg.83? n
mget uuu.8? y
200 PORT command successful.
150 Opening ASCII mode data connection for uuu.8.
226 Transfer complete.
1793 bytes received in 1.355 seconds (1.292 Kbytes/s)
ftp> get ggg.22
200 PORT command successful.
550 *** ERROR *** NO BATCHES FOR TRANSMISSION
ftp>
```

Chapter 3

FTP $$ Commands

Connect:Enterprise accepts commands in two formats: Mailbox ($$) syntax and standard FTP syntax.

This chapter describes the commands available to FTP remote users using $$ syntax. Refer to Chapter 2, *Standard FTP Commands*, for information about standard FTP commands.

- **delete** or **del**—Flags a batch of data as deleted.
- **dir**—Requests a formatted listing of batches from the host site.
- **get**—Requests a batch of data from the host site.
- **mget**—Requests multiple batches of data from the host site.
- **put**—Sends a batch of data to the host site.

The **user**, **binary** (**bin**), **ascii**, **prompt**, and **runique** commands are also available to users of Connect:Enterprise. Before using any commands, ensure that you have met the prerequisites listed in the following section.

Requirements

Obtain the following information from your host site system administrator prior to executing a Connect:Enterprise command:

- Mailbox ID
- Password and any additional site-specific security, including information about the *mbxacl.conf* file that restricts users from performing specific commands in specific mailbox IDs other than their own.
- IP address or host name and FTP port
- Secure FTP requirements
- To use Secure FTP, both the sending and receiving sites must have FTP software that is SSL capable.

Logging On to Connect:Enterprise

If your remote site uses FTP communications software, use the commands and procedures that follow to access and use Connect:Enterprise:

1. At the command line prompt for your system (shown as $ here), type the following command and parameters:

```
$ ftp host_name port_number
```

The following table describes the available parameters.

Parameter	Description
host_name	The name of the system running Connect:Enterprise. You can enter the IP address of the host instead of the host name.
port_number	The Connect:Enterprise FTP port listener number. Refer to the Define System functionality of the Site Administration user interface to specify this port FTP port.

Because some FTP clients do not allow arguments on the FTP command, you can also access Connect:Enterprise by typing the **FTP** command without arguments, followed by the **open** command at the ftp> prompt, as shown in the following example:

```
$ ftp
ftp> open host_name port_number
```

The host site prompts you for a login ID, as shown in the following example.

```
Name (host_name:default_userid): logon_ID
```

2. Enter a logon ID. This must be defined in an account definition by a Connect:Enterprise site administrator. If a password is required for your site, it also must be entered in the account definition.

   You are prompted for a password, as shown in the following example:

```
Password:
```

3. Enter the password.

   The following is an example of a remote user on a workstation called *bevo*, logging into the Connect:Enterprise on a local (host) workstation called *local*, where:

   - The command line prompt on the remote system is bevo>.
   - The host_name argument to the **FTP** command is *local.*

- The port_number argument is 10031.
- The user name (logon_ID) is mary.
- The default_userid (steve) is different from the logon_ID (mary).

If the logon_ID matches the default_userid, pressing **Enter** is sufficient. In this example, the default_userid (steve) is different from the logon_ID (mary). As you enter the password, it is not displayed. User inputs are shown in bold.

```
bevo> FTP local 10031
Connected to local.group.company.com.
220 <<<Connect:Enterprise 2.1.00>>> at local FTP server ready.
Time = 16:02:42
Name (labs:steve): mary
331 Password required for mary.
Password:
230 Connect:Enterprise login ok, access restrictions apply.
ftp>
```

You have now accessed Connect:Enterprise.

After you are connected to the host site system, you can send data to and receive data from the repository with the commands described in the following sections.

Changing Your Password

There are two messages associated with changing your password:

- If your password is about to expire, you will get a message similar to the following:

```
230 Connect:Enterprise UNIX login ok, access restrictions apply, password expires
in 1 day.
```

In this case, the next time you log on, type the following at the password prompt:

```
Password: oldpassword/newpassword/newpasssword
```

Where *oldpassword* is your expired password and *newpassword* is the password you want to change it to.

- If your password is expired, your log on attempt will fail with a message similar to the following:

```
539 User $$user01 login incorrect, password change required
Login failed.
```

In this case, attempt to log on again. At the password prompt, type the following:

```
Password: oldpassword/newpassword/newpasssword
```

Where *oldpassword* is your expired password and *newpassword* is the password you want to change it to.

Logging Off FTP

When you are finished using the Connect:Enterprise FTP server, you must log off using one of two FTP commands: **bye** or **quit**. In the following example, the user enters **quit**, receives a 221 message (Goodbye), and returns to the command line prompt (bevo>).

```
ftp> quit
221 Goodbye.
bevo>
```

delete Command

This command flags a specific batch of data as logically deleted from your mailbox, by setting the deleted (D) flag.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file. The contents of this file determine the mailboxes you can delete batches from. If you enter a **delete** command for a mailbox you do not have access to, the message *Permission denied* is returned.

The command can be abbreviated to **del**. After Connect:Enterprise receives the **del** command, it processes the command and returns the results to the remote site. The remote site collects the output to the console.

When a batch is logically deleted, the D flag is set for that batch. The host site operator can restore the batch using the **cmustatus** command or physically erase it using the **cmuerase** command.

del Command Format

The following example shows the $$ syntax for the **del** command:

```
ftp> del "$$ parameters"
```

$$ delete Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both kinds of parameters are listed alphabetically.

With the $$ syntax, you can use any number of blanks between parameters, but blanks are not allowed within parameters, except for the batch ID. All text following the **del** command (including $$) must be enclosed in one set of double quotation marks.

Command	Parameter
delete	**$$**
	BID='xx...xx'|#nnnnnnnn
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ORIG=XXXXXXXX
	PASSWORD=XXXXXXXX
	RECEIPT
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the parameters. Required parameters are in bold.

Parameter	Description
BID='xx..xx'| #nnnnnnnn	indicates either the user batch ID or the batch number. The user batch ID can be 1–64 characters. Wildcard specifications (like an asterisk, *) are supported. A single-word batch ID does not require quotation marks. To specify a multiword BID with spaces, enclose the BID in single quotation marks (' '). To specify a batch number, precede the batch number with a pound sign (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign, and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.

Parameter	Description
FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I deletes all batches that were added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I deletes the batches that are neither added locally nor incomplete. The syntax CM!I deletes batches that are collected remotely, are multitransmittable, and not incomplete.
FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for deletion. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **delete** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **delete** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 deletes all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. FTIME=020101:1315 limits the **delete** command to those batches created on or after January 1, 2002 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow. The status report indicating which batches were logically deleted is written to standard output. If an error is encountered when the command line is parsed, the error messages are written to standard error.
ID=xxxxxxxx	indicates the valid mailbox ID of the site requesting the deletion and can be 1–8 characters. Wildcard specifications (like an asterisk, *) are supported.

Parameter	Description
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.
RECEIPT	resets the log maintained in the log batch called the <<ACTIVITY LOG>>. This parameter cannot be used with any other optional parameter.
TTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on which or before which a batch must have been created to be eligible for deletion. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **delete** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **delete** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 TTIME=20020101 deletes all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **delete** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow. The status report indicating which batches were logically deleted is written to standard output. If an error is encountered when the command line is parsed, the error messages are written to standard error.

Note: The **del** command can have an unexpected result when used under certain conditions. Refer to the *Considerations When Using the del, mdel, get, and mget Commands* on page 39 for more information on how to avoid this potential problem.

$$ delete Examples

In the following example, ACME deletes a specific batch by specifying its batch number. ID is optional for processing the **delete** command. This command logically deletes batch number 775 and sets the deleted (D) flag. Deleted batches are not available for transmission, even if the requestable flag is on, and a specific batch number is specified on the request. Deleted batches can be extracted at the host site, if the specific batch number is specified with the **cmuextract** command. Batches can be restored only at the host site with the **cmustatus** command.

```
ftp> del "$$ ID=ACME BID=#775"
250 DELE command successful.
ftp>
```

In the next example, ACME narrows the deletion of ACME batches to those that have the user batch ID 'invoices' as well as the mailbox ID ACME.

```
ftp> del "$$ ID=ACME BID='invoices'"
```

In the following example, ACME deletes the batch indicating the receipt of data (the log batch, called <<ACTIVITY LOG>>). For the receipt batch only, the **delete** command erases it completely. This use of the **delete** command with $$ syntax is applicable only to the log batch.

After the log batches are erased, the next add or extract of batches creates a new log batch for the remote user who initiated the add or extract and whose account definition has LOGBATCH=Y specified. The space between $$ and RECEIPT is optional.

```
ftp> del "$$RECEIPT"
```

dir Command

The **dir** command displays information about batches in the current working mailbox.

Note: The **ls** command can be used in place of **dir**.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file. These permissions determine the mailboxes you can obtain directory listings for. If you enter a **dir** command for a mailbox you do not have access to, you get the message *Permission denied*.

Batches can be selected (filtered) is by mailbox ID alone, mailbox ID and batch ID, by batch number, originator, flags, and start and end times.

The contents are displayed either to standard output or to a local file if one is specified as an argument.

dir Command Format

The following shows the $$ syntax for the **dir** command:

```
ftp> dir "$$ parameters" [local_filename]
```

$$ dir Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both kinds of parameters are listed alphabetically.

You can use any number of blanks between parameters, but not within parameters, except for the BID. A BID with spaces must be enclosed in single quotation marks. All text following the **dir** command (including $$) must be enclosed in one set of double quotation marks.

Command	Parameter
dir	**$$**
	BID='xx...xx'|#nnnnnnnn
	BCHSEP=NO**|**OPT3
	dest_filename
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ONEBATCH=YES|Y|NO|N
	ORIG=XXXXXXXX
	RECEIPT
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the parameters. Required parameters are in bold.

Parameter	Description
$$	indicates that mailbox (nonstandard FTP) syntax is being used.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the mailbox.

Parameter	Description
BID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. The user batch ID can be 2–64 characters. It must be enclosed in single quotation marks if it contains spaces. Wildcard specifications (like an asterisk, *) are allowed. If a specific batch number is specified, a pound sign must precede the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.
FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11. You can negate a flag by adding an exclamation point (!) in front of it. For example, A!I requests a directory listing for all batches that were added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I requests a directory listing of the batches that are neither added locally nor incomplete. The syntax CM!I specifies to list batches collected remotely, that are multitransmittable, and are not incomplete.

Parameter	Description
FTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for reporting. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME= 990101 limits the **dir** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **dir** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. FTIME=020101:1315 limits the **dir** command to those batches created on or after January 1, 2002 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for reporting. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

Parameter	Description
TTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on or before which a batch must have been created to be eligible for reporting. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **dir** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **dir** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. TTIME=011231:1400 limits the **dir** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for reporting. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID directory information is desired for. Connect:Enterprise searches the repository for all batches that match the specified mailbox ID. Wildcard specifications (like an asterisk, *) are supported.
local_filename	identifies the destination file name where the batch information records of the mailbox are written. Include pathnames when necessary. When a file name is not specified, the batch information records are returned to the remote site's display. If no file name is used, output can still be collected to a file by conducting the entire FTP session from a script file with output redirected to a specified file.

Output

After a **dir** command is issued, Connect:Enterprise scans the repository for the designated batches and returns that information to the remote site in the following format:

```
xxxxxxxx nnnnnnnn bbbbbbbbbb <x...x> YYMMDD-HHMM flags protocol format
```

The following table describes the output.

Section	Description
xxxxxxxx	The 8-character mailbox ID for the batch.
nnnnnnnn	The 8-digit batch number assigned to the batch.
bbbbbbbbbb	The total byte count of the batch.
<x...x>	The first 24 characters of the batch's user batch ID. If the user batch ID is longer than 24 characters, it is shortened to the first 23 characters of the literal with the 24th character becoming a plus sign (+).
YYMMDD–HHMM	The date and time that the batch was added to the repository.
flags	One or more flags identifying the current status of a batch in the repository. Refer to *Flags* on page 11 for definitions of these flags. ◆ TCP Transmitted locally with **cmuadd** ◆ ASY Transmitted remotely with Async protocol ◆ BSC Transmitted remotely with Bisync protocol ◆ FTP Transmitted remotely with FTP protocol ◆ FTS Transmitted remotely with Secure FTP protocol ◆ (blank) Activity log batch
protocol	indicates the transmission protocol used to place the batch into the specified mailbox.
format	indicates the data format of the batch. Valid values are ASC (ASCII), BIN (binary), or EBC (EBCDIC).

$$ Syntax dir Examples

In the following example, ACME generates a listing of all batches stored in the repository whose mailbox ID is ACME. With no destination file name specified, the batch information records are displayed on screen. If ACME is the remote site entering these commands, then to obtain batches in the repository called ACME, specifying ID=ACME is optional.

```
ftp> dir "$$ ID=ACME"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ iD=ACME.
ACME    00000020 00000152 <test batch>       990926-1431 CD   FTP ASC
ACME    00000019 00000152 <test batch>       990926-1435 CD   FTP ASC
ACME    00000014 00000606 <cmuadded locally> 990929-1306 A R TCP BIN
ACME    00000013 00000606 <orders>           990929-1308 A R TCP EBC
ACME    00000004 00000606 <orders>           990929-1309 A R TCP ASC
Total Number of batches listed: 5
226 Transfer complete.
ftp>
```

The next example illustrates how ACME narrows the selection of batches to those with the user batch ID *orders*. All batch information records are returned to ACME as a single, concatenated file named *batinfo.dat*.

```
ftp> dir "$$ ID=ACME BID='orders'" batinfo.dat
output to local-file: batinfo.dat? y
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=ACME BID='orders'.
226 Transfer complete.
ftp>
```

In the following example, ACME views the characteristics and status of a specific, previously obtained batch number. The batch status information is collected in the destination file *batch13.dat*.

```
ftp> dir "$$ ID=ACME BID=#13" batch13.dat
output to local-file: batch13.dat? y
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=ACME BID=#13.
226 Transfer complete.
ftp>
```

In the next example, ACME uses the default for the ID and PASSWORD parameters, that is, those specified when the FTP user command was issued to log on. Batch information records are returned for all mailbox batches with a mailbox ID matching the login ID. In this case, the remote user is *steve*, so the output shown is for the mailbox with the ID *steve*.

```
ftp> dir "$$"
200 PORT command successful.
150 Opening ASCII mode data connection for $$.
steve   00000005 00000302 <a batch id>       990925-1434  A RT TCP ASC
steve   00000024 00000302 <a batch id>       990925-1436  A RT TCP ASC
steve   00000022 00000302 <a batch id>       990925-1440  A RT TCP ASC
steve   00000021 00000302 <New stuff>        990925-1617  A RT TCP ASC
steve   00000018 00000606 <cmuadded locally> 990929-1259  A R  TCP ASC
steve   00000017 00000606 <cmuadded locally> 990929-1301  A R  TCP BIN
Total Number of batches listed: 6
226 Transfer complete.
ftp>
```

The following example shows the use of the ORIG option. The command returns all batches whose originator ID is the *originator name* indicated (within the constraints set forth by the system administrator in the *mbxacl.conf* file). In other words, if the current directory has no batches available to the current user (based on that user's permissions), then the **dir** command does not return any batches.

Note the following three batches (batch numbers 20, 14 and 13). These are a subset of the batches shown in the first example. In that example, the output included two batches with the batch ID *test batch* and two more with the batch ID *orders*. This indicates to the user that the two other batches (batch numbers 19 and 4) were not originated by *steve*.

```
ftp> dir "$$ ID=ACME ORIG=steve"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=ACME ORIG=steve.
ACME   00000020 00000152 <test batch>       990926-1431  CD   FTP ASC
ACME   00000014 00000606 <cmuadded locally> 990929-1306  A R  TCP BIN
ACME   00000013 00000606 <orders>           990929-1308  A R  TCP EBC
Total Number of batches listed: 3
226 Transfer complete.
ftp>
```

The following example give a directory listing of all batches originated by *dave* at or after September 29, 2002, 1:25 pm.

```
ftp> dir "$$ ORIG=dave FTIME=20020929:1325"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ORIG=dave FTIME=20020929:1325.
dave   00000023 00000152 <steve>           990929-1327  CD     FTP ASC
dave   00000016 00000152 <batch by dave>   990929-1327  C      FTP ASC
dave   00000015 00000069 <next dave batch> 990929-1328  C      FTP ASC
Total Number of batches listed: 3
226 Transfer complete.
ftp>
```

The **Flags** parameter is a filtering identifier. The next example is identical to the preceding one, except for the Flags=!D. Note that unlike the preceding example, batch 23 (with the D flag set) is not listed.

```
ftp> dir "$$ ORIG=dave FTIME=20020929:1325 FLAGS=!D"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ORIG=dave FTIME=20020929:1325
FLAGS=!D.
dave    00000016 00000152 <batch by dave>   990929-1327  C     FTP ASC
dave    00000015 00000069 <next dave batch> 990929-1328  C     FTP ASC
Total Number of batches listed: 2
226 Transfer complete.
ftp>
```

The following example returns all batches that are flagged for multiple transmission (M) and are requestable (R).

```
ftp> dir "$$ FLAGS=MR"
```

get Command

The **get** command requests a single file from the repository. This single file contains all the batches matching the selection criteria. For example, if ID=j* is specified, all batches contained in mailboxes with names beginning with j are sent as one file.

Note: The **recv** command is synonymous with the **get** command and these can be used interchangeably.

The **get** command copies a file from the repository to the local directory. Multiple batches with the same batch ID are always concatenated into a single file, no matter what BCHSEP is set to. In order to get the batches individually (rather than concatenated), you must use the **mget** command.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file. This file indicates where you can retrieve batches from.

If the data format is EBCDIC or binary, the remote user must enter the FTP command **bin** before issuing the **get** command. For ASCII data format enter the **asc** FTP command.

get Command Format

The following example shows the $$ syntax for the **get** command:

```
ftp> get "$$ parameters" [dest_filename]
```

Note: The **get** command can have unexpected results under certain conditions. Refer to the *Considerations When Using the del, mdel, get, and mget Commands* on page 39 for more information on how to avoid this potential problem.

$$ get Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both types of parameters are listed alphabetically.

The keyword is followed by double quotation marks, two dollar signs, any parameters needed, and an ending double quotation marks. The dest_filename must be the last option after the final double quotation marks. The double quotation marks are required. Refer to the syntax example in the *get Command Format* section on *get Command Format* on page 58.

You can use any number of blanks between parameters, but not within parameters, except for the BID, which must be enclosed in single quotes if spaces are embedded.

Command	Parameter
get	$$
	BID='xx...xx'|#nnnnnnnn
	BCHSEP=NO|OPT3

Command	Parameter
	CONV=N|A|E
	dest_filename
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ONEBATCH=YES|Y|NO|N
	ORIG=XXXXXXXX
	RECEIPT
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
$$	indicates that mailbox (nonstandard FTP) syntax is being used.
BID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. It can be 1–64 characters. To create a multiword BID, enclose the user batch ID in single quotation marks (' '). Single-word BIDs (with no spaces) do not require quotes. Wildcard specifications (like an asterisk, *) are supported. To specify a batch number, use a pound sign followed by the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.
BCHSEP=NO|OPT3	specifies the FTP transmission format to send output to the remote site. Options are: **NO|N—**transmits all batches as a single concatenated file. Each batch is flagged as transmitted (T) immediately after it is sent. This is the default. **OPT3**—transmits all batches as a single concatenated file. Successfully transmits all batches before it flags any batches as transmitted.

Parameter	Description
CONV=N|A|E	specifies the transmission code set in which the remote site receives data. Because some conversions are not allowed (such as binary to ASCII), this parameter can limit the batches selected for transmission. CONV processing rules can be specified in the account definition and overridden in the remote block of a schedule. If no **CONV** parameters (at any level) are supplied, the default is CONV=N. **N**—Specifies that no translation is to occur. All batches that meet the **get** requirements are transmitted to the remote site without translation. This is the default. **A**—Specifies to transmit the ASCII data to the remote site. When A is specified, batches marked as containing ASCII or EBCDIC data and meeting the **get** requirements are transmitted to the remote site. EBCDIC batches that are selected are translated to ASCII based on user or Sterling Commerce-supplied translation tables (specified by the account definition or remote block in a schedule). For binary conversions, no conversion is done. **E**—Specifies to transmit EBCDIC data to the remote site. When E is specified, batches marked as containing ASCII or EBCDIC data and meeting the **get** requirements are transmitted to the remote site. ASCII records are translated to EBCDIC through user or Sterling Commerce-supplied translation tables.
dest_filename	identifies the destination file name and directory path name for the data you want from the mailbox. Depending on the file system being used, the **get** command can truncate file names (for example, to the 8.3 format on DOS). Therefore, always supply the dest_filename argument if it is known.
FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I specifies to get all batches added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I specifies a **get** request the batches that are neither added locally nor incomplete. The syntax CM!I specifies a **get** request for batches that were collected remotely, are multitransmittable, and are not incomplete.

Parameter	Description
FTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for retrieval. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **get** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **get** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. Specifying FTIME=010101:1315 limits the **get** command to batches created on or after January 1, 2002 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID of the remote site requesting the data. Wildcard specifications (like an asterisk, *) are supported. If neither the **BID** nor the **ID** parameter is specified, all batches in the current mailbox are selected and concatenated into one file.
ONEBATCH=YES|Y|NO|N	indicates that only the first batch matching the selection criteria is to be transmitted to the remote site in response to the **get** command. **YES|Y**—Specifies to transmit only a single batch to the remote site. This is especially useful with the **mget** command: when a specific batch number has not been specified and you want to prevent concatenation of multiple batches by limiting the transmission to one batch, use this parameter. **NO|N** —Specifies to transmit all batches that conform to the request to the remote site. This is the default.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.

Parameter	Description
RECEIPT	sends the log batch called <<ACTIVITY LOG>>. It is a log of all the ADD/REQ activity performed by the remote user, and is created if the account definition parameter **Log receipt of batch** is checked. This batch can be used as a receipt by the remote user. This batch is reset by the command **del "$$receipt"**. This parameter cannot be used with any other optional parameter.
TTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on or before which a batch must have been created to be eligible for retrieval. If the **FTIME** parameter is omitted, the creation date and time of oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **get** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **get** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20000101 lists all batches created on or between December 31, 1999 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. The entry TTIME=011231:1400 limits the **get** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ Option—Description ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

$$ get Examples

Failing to supply the required output file name when issuing a **get** command can create a file in the current directory with a name equal to the string contained with the double quotation marks portion of the **get** command. This file contains the data you intended to obtain from the Connect:Enterprise host. However, the unusual file name can cause problems when you attempt to view, copy, move, or even remove the file. For example, when you issue the following command:

```
get "$$ ID=dave"
```

After ending your FTP session, you find a file in the current directory named:

```
$$ ID=dave
```

The following example shows the situation described in the preceding example. During the FTP session, you enter the **get** command. Then you issue the **quit** command to exit the session and return to the shell prompt. Next, the **ls -al** command displays files in the local directory, including a file called *$$ ID=dave*.

```
ftp> get "$$ ID=dave"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=dave.
226 Transfer complete.
228 bytes received in 0.4353 seconds (0.5115 Kbytes/s)
ftp> quit
221 Goodbye.
mynode> ls -al
total 72
-rw-r--r--   1 steve    develop      221 Sep 29 13:43 $$ ID=dave
drwxr-xr-x   2 steve    develop      512 Sep 29 13:43 ./
drwxr-xr-x  17 steve    develop      512 Aug 27 16:11 ../
-rw-r--r--   1 steve    develop       72 Sep 25 16:09 ACME
-rw-r--r--   1 steve    develop       69 Sep 29 13:27 dave
-rw-r--r--   1 steve    develop     5949 Oct  9 1996  sample_rsd
-rw-r--r--   1 steve    develop      152 Sep 25 14:39 steve
mynode>
```

The $$ and space characters can cause problems. They must be prefixed with the back slash (\) when the file name is referenced on the command line. The following example renames the file to *output.dat*.

```
mv   \$\$\ ID=dave  output.dat
```

A space follows the third back slash (\) in the example.

In the first example, ACME retrieves all the eligible batches that have been added to the repository at the host site. Eligible batches are batches that have a requestable flag but do not yet have a transmitted (T), nontransmittable (N), or deleted (D) flag. ACME can view these status flags by first issuing a **dir** command to receive a listing of the ACME batches. Having taken the default parameter, CONV=N, no transmission code conversions are done. Batches of all types (ASCII, EBCDIC, and binary) are retrieved.

This FTP **get** command collects one or more eligible batches as a single concatenated file named *receive.dat.* Because the default batch separation method BCHSEP=NO is in effect, as each batch in the concatenation is successfully transmitted, Connect:Enterprise immediately flags it as transmitted. Although the requestable flags are retained after transmission, if ACME issues this

same **get** command in the future, those batches are no longer eligible. However, newly added batches, lacking transmitted flags, are eligible with one exception: If the batch is originally added to the repository with a MULTXMIT=Y parameter, it has a multitransmittable flag that prevents the batch from ever being flagged as transmitted. This leaves it eligible for transmission to other sites or to the same site multiple times.

```
ftp> get "$$ ID=ACME" receive.dat
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=ACME.
226 Transfer complete.
619 bytes received in 0.5157 seconds (1.172 Kbytes/s)
ftp>
```

ACME can use the **get** command shown in the preceding example to request a specific batch, having previously obtained batch numbers from a **dir** report. This command can be used to force retransmission of a previously transmitted batch or to select a unique batch. This command does not permit collection of a deleted batch nor a nontransmittable batch. It does not permit collection of a batch that does not have a requestable (R) flag.

The next example shows the use of the CONV=A parameter. The requested batch is converted to ASCII if it is flagged as EBCDIC; it is left unconverted if it is already flagged as ASCII; or it is not is transmitted at all if it is flagged BINARY (because binary data cannot be converted to ASCII). If the batch does not have a transmitted (T) flag, it is flagged as transmitted after successful transmission, and the requestable (R) flag is retained.

However, there is an exception: If the batch is added to the mailbox with a MULTXMIT=Y parameter, it has a multitransmittable (M) flag that prevents it from being flagged as transmitted. This leaves it eligible for transmission to other sites or to the same site multiple times.

```
ftp> get "$$ ID=ACME BID=#25 CONV=A" batch25.dat
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=ACME BID=#25 CONV=A.
226 Transfer complete.
179 bytes received in 0.2435 seconds (0.7179 Kbytes/s)
ftp>
```

In the next example, ACME requests a subset of all the batches that are eligible for transmission. By specifying a user batch ID, ACME limits the retrieved batches to those with the user batch ID of *invoices*.

Because the remote site, ACME, has been receiving all the batches as a single file, they do not want to dial back into the host site and use the **get** command to collect the remaining untransmitted batches. ACME retries the whole transmission and retrieves all the batches from the beginning. To avoid this problem, ACME has chosen to concatenate the batches by using BCHSEP=OPT3. This parameter instructs Connect:Enterprise to wait until all eligible batches have been successfully transmitted before flagging them all as transmitted. If the transmission fails on the second or subsequent batch in a concatenation, a subsequent, identical **get** command collects all the batches.

With BCHSEP=NO, the batches are sent concatenated but they are flagged as transmitted as each batch is transmitted. They are collected to the single file, *invoices.dat*. When BCHSEP=NO is used,

the risk of batches being flagged as transmitted early in a transmission that later fails before all the expected batches are transmitted exists. Batches transmitted successfully are flagged as transmitted and are no longer available.

```
ftp> get "$$ ID=ACME BID=invoices BCHSEP=OPT3" invoices.dat
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=ACME BID=invoices BCHSEP=OPT3.
226 Transfer complete.
358 bytes received in 0.4394 seconds (0.7956 Kbytes/s)
ftp>
```

ACME logged on to the FTP server with a mailbox ID and password. In the next example the, **get** command collects all batches with that same ID added to the repository. Default values for all other request parameters are assumed as well.

```
ftp> get "$$" allfiles.dat
200 PORT command successful.
150 Opening ASCII mode data connection for $$.
226 Transfer complete.
358 bytes received in 0.6192 seconds (0.5646 Kbytes/s)
ftp>
```

In the following example, ACME requests the <<ACTIVITY LOG>> batch maintained by Connect:Enterprise for ACME. This batch contains the log of all ADD and REQ operations performed by ACME. This batch is maintained only if Connect:Enterprise has enabled the verification of data delivery feature for the remote site by checking the **Create log batch?** parameter in the ACME account definition.

To prevent confusion with other batches, explicitly request a receipt by indicating:

```
ftp> get "$$receipt"
200 PORT command successful.
150 Opening ASCII mode data connection for $$receipt.
226 Transfer complete.
1859 bytes received in 0.6925 seconds (2.622 Kbytes/s)
ftp>
```

The next example is similar to the preceding example, but the <<ACTIVITY LOG>> batch is extracted to a file called *xxx*.

```
ftp> get "$$receipt" xxx
200 PORT command successful.
150 Opening ASCII mode data connection for $$receipt.
226 Transfer complete.
2077 bytes received in 0.7651 seconds (2.651 Kbytes/s)
ftp>
```

The following command returns all batches created between the FTIME and the TTIME time stamps and concatenates them into one file called *local filename*.

```
ftp> get "$$ FTIME=19990929:1300 TTIME=990929:1500" "local filename"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ FTIME=19990929:1300 TTIME=990929:1500.
226 Transfer complete.
358 bytes received in 0.6206 seconds (0.5634 Kbytes/s)
ftp>
```

If you do not enclose *local filename* in double quotes, then the file created on the remote system is called *local* instead of *local filename*.

Unavailable Batches

Batches are unavailable in the following circumstances:

- Batches flagged as nontransmittable are not available to Remote Connect **$$REQUEST** or **get** commands.
- Batches flagged as deleted (D) are not available to Remote Connect **$$REQUEST** or **get** commands.
- Batches lacking the requestable (R) flag are not available to Remote Connect **$$REQUEST** or **get** commands. These were collected from a remote site that did not use XMIT=Y, MULTXMIT=Y, or TO=Y on the **$$ADD** card and thus, are not eligible for retransmission.
- Batches flagged as transmitted (T) are not available to Remote Connect **$$REQUEST** or **get** commands unless a batch number is indicated.
- Batches flagged as incomplete are not available to Remote Connect **$$REQUEST** or **get** commands unless a batch number is indicated.
- Binary batches are not available when CONV=A or CONV=E is specified. Binary batches cannot be converted to ASCII or EBCDIC.
- Batches that do not match the specified mailbox ID and/or user batch ID are not available.

mget Command

The **mget** command results in a file being created for each batch matching the selection criteria. These discrete files are saved under the name *batchno.dat*, where *batchno* is the mailbox batch number.

The permissions set for your mailbox by the system administrator in the *mbxacl.conf* file determine where you can retrieve batches from.

All Connect:Enterprise remote connects using FTP **mget** must be preceded by the FTP command **binary** if use of CONV=A or E is anticipated. This command turns off the FTP client's translation functionality at the remote site, permitting the server, Connect:Enterprise, to have complete control over conversion of data formats through the user-specified **CONV=** parameter. All formats (binary, ASCII, and EBCDIC) can be transmitted with binary mode specified at the remote site.

mget Command Format

The following shows the $$ syntax for the **mget** command

```
ftp> mget "$$ parameters"
```

Note: The **mget** command can have unexpected results when used under certain conditions. Refer to the *Considerations When Using the del, mdel, get, and mget Commands* on page 39 for more information on how to avoid this potential problem.

$$ mget Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both types of parameters are listed alphabetically.

With the $$ syntax, you can use any number of blanks between operands, but blanks cannot not be used within parameters, except for batch ID. All text following the **mget** command (including $$) must be enclosed in one set of double quotation marks.

Command	Parameter
mget	BID='xx...xx'|#nnnnnnnn
	CONV=N|A|E
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ORIG=XXXXXXXX
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
$$	indicates that the following data is destined for the mailbox.

Parameter	Description
BID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. It can be 1–64 characters. To create a multiword BID, enclose the user batch ID in single quotation marks (' '). Single-word BIDs (with no spaces) do not require quotes. Wildcard specifications (like an asterisk, *) are supported. To specify a batch number, place a pound sign in front of the batch number (for example, #14). The batch number can be up to eight digits. Leading zeroes are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign. If neither the **BID** nor the **ID** parameter is specified, all batches in the current mailbox are selected and you are prompted to save each batch to a separate file.
CONV=N|A|E	specifies the transmission code set in which to transmit data to the remote site. Because some conversions are not allowed (such as binary to ASCII), this parameter can limit the batches selected for transmission. For this parameter to function correctly, **Outbound batch separation** parameter must be set to **Batches are created as individual files...** in the account definition. CONV processing rules can be specified in the account definition and overridden in the remote block of a schedule using the **Outbound data format** parameter. If no **CONV** parameters (at any level) are supplied, the default is CONV=N. **N**—Specifies that no translation is to occur. All batches that meet the **mget** command requirements are transmitted to the remote site without translation. This is the default value. **A**—Specifies transmission of ASCII data to the remote site. When A is specified, batches marked as containing ASCII or EBCDIC data and meeting the **mget** requirements are transmitted to the remote site. EBCDIC batches that are selected are translated to ASCII based on user supplied or Sterling Commerce-supplied translation tables (specified in the account definition or remote block of a schedule). **E**—Specifies transmission of EBCDIC data to the remote site. When E is specified, batches marked as containing ASCII or EBCDIC data and meeting the **mget** requirements are transmitted to the remote site. ASCII records are translated to EBCDIC through user-supplied or Sterling Commerce-supplied translation tables.

Parameter	Description
FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of all flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I retrieves all batches that were added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I specifies to get the batches that are neither added locally nor incomplete. The syntax CM!I in a **get** request specifies batches collected remotely that are multitransmittable and not incomplete.
FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for retrieval. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **mget** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **get** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. TTIME=011231:1400 limits the command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID of the remote site requesting the data. Wildcard specifications (like an asterisk, *) are allowed. If neither the **BID** nor the **ID** parameter is specified, all batches in the current mailbox are selected and you are prompted to save each batch to a separate file.

Parameter	Description
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the mailbox.
TTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on or before which a batch must have been created to be eligible for retrieval. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **mget** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **get** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter an be further defined using the :hhmm subparameter. The entry TTIME=011231:1400 limits the **mget** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

$$ mget Examples

In the following example, the remote site uses the **mget** command to collect data into multiple files. ACME has used CONV=E to limit the selection of batches to those that are already in EBCDIC format and ASCII batches to be converted to EBCDIC, if any. All data transmitted from the host site must be in EBCDIC format. No binary data is collected. Only batches flagged ASCII or EBCDIC are eligible.

The request limits the criteria further to only those batches selected by ID=ACME and CONV=E. When BCHSEP is NONE or OPT3, all batches are concatenated into a single file. When BCHSEP=OPT4 is specified in the remote user's account definition, as is the case in this example, Connect:Enterprise assumes that the batches are created as separate, individual files when the **mget** command is used. Therefore, one or more files matching the criteria defined here are collected at

the specified remote site. Each file has a unique name in the format *batch_id.batchno*, where *batchno* is the mailbox batch number.

In this example, the user first enters a **dir** command to show the characteristics of the batches of interest. Batch numbers 28 and 29 in the ACME mailbox have the batch ID *orders*. Batch 28 is not requestable (its R flag is not set), so on the subsequent **mget** command, the transmission for it fails. However, batch 29 is requestable, and is an EBCDIC batch, so the transmission succeeds.

```
ftp> dir "$$ ID=ACME bid=orders"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ bid=orders.
ACME     00000028 00000174 <orders>     990929-1519  C         FTP ASC
ACME     00000029 00000174 <orders>     990929-1519  C R  M    FTP EBC
Total Number of batches listed: 2
226 Transfer complete.
ftp> bin
200 Type set to I.
ftp> mget "$$ ID=ACME BID=orders CONV=E"
mget orders.28? y
200 PORT command successful.
550 *** ERROR *** NO BATCHES FOR TRANSMISSION
mget orders.29? y
200 PORT command successful.
150 Opening BINARY mode data connection for orders.29.
226 Transfer complete.
174 bytes received in 0.2508 seconds (0.6775 Kbytes/s)
ftp>
```

Because the remote user requests EBCDIC batches, the mode must first be set to binary with the **bin** command. If the mode were ASCII, the **mget** command that succeeded in the preceding example (for batch 29), would have failed.

In the following example, the **mget** command returns all requestable batches whose originator ID is *steve* indicated (within the constraints set forth by the system administrator in the *mbxacl.conf* file). In other words, if the current directory has no batches available to the current user (based on that user's permissions), then the **mget** command does not return any batches.

In this example, originator *steve* has put several batches into the ACME mailbox (with four different batch IDs). The **Outbound batch separation** parameter is not defined in the ACME account definition, so no batch separation occurs. Therefore, none of the batches originated by *steve* with batch IDs *cmuadded locally* or *orders* are requestable (thus the error messages), but the batches originated by *steve* with the batch IDs *example* and *invoices* are available for retrieval.

```
ftp> mget "$$ ORIG=steve"
mget cmuadded locally? y
200 PORT command successful.
550 *** ERROR *** NO BATCHES FOR TRANSMISSION
mget orders? y
200 PORT command successful.
550 *** ERROR *** NO BATCHES FOR TRANSMISSION
mget p. 5-31 example? y
200 PORT command successful.
150 Opening ASCII mode data connection for p. 5-31 example.
226 Transfer complete.
179 bytes received in 0.2939 seconds (0.5947 Kbytes/s)
mget invoices? y
200 PORT command successful.
150 Opening ASCII mode data connection for invoices.
226 Transfer complete.
179 bytes received in 0.3049 seconds (0.5734 Kbytes/s)
ftp>
```

The following example retrieves all available batches originated by *dave* at or after September 29, 2002 at 1:25 pm. At least one batch with batch ID *another batch* is requestable, but none of the batches with the batch ID *dave account definition* are available; thus the error message is returned.

```
ftp> mget "$$ orig=dave FTIME=20020929:1325"
mget another batch? y
200 PORT command successful.
150 Opening ASCII mode data connection for another batch.
226 Transfer complete.
156 bytes received in 0.2446 seconds (0.6228 Kbytes/s)
mget dave rsd file? y
200 PORT command successful.
550 *** ERROR *** NO BATCHES FOR TRANSMISSION
ftp>
```

The **flags** parameter is a filtering identifier. In the following example, note that the user is prompted for the batch that is not requestable (which results in an error message), because the FTP client makes each request, whereas the Connect:Enterprise server detects the failure and generates the error message.

```
ftp> mget "$$ orig=dave FTIME=20020929:1325 flags=R"
mget another batch? y
200 PORT command successful.
150 Opening ASCII mode data connection for another batch.
226 Transfer complete.
156 bytes received in 0.2446 seconds (0.6228 Kbytes/s)
mget dave rsd file? y
200 PORT command successful.
550 *** ERROR *** NO BATCHES FOR TRANSMISSION
ftp>
```

The following example returns all batches that are flagged for multiple transmission (M) and are requestable (R).

```
ftp> mget "$$ FLAGS=MR"
```

put Command

The **put** command adds a single batch of data to the current working mailbox. If the data format is EBCDIC or BINARY, the remote user must enter the FTP command **bin** before issuing the **put** command. For ASCII data format, enter the **asc** FTP command. The user-specified parameters control characteristics of the batch being sent.

The permissions are set for your mailbox by the system administrator in the *mbxacl.conf* file. The contents of this file determine the mailboxes to which you can add batches with the **put** command.

put Command Format

The following example shows the $$ syntax for the **put** command:

```
ftp> put source_path "$$ parameters"
```

$$ put Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font.

The **source_path** parameter must be the first option, immediately after the **put** command. This is followed by a double quotation mark, two dollar signs, any parameters needed and an ending double quotation mark. The double quotation marks are required.

Command	Parameter
put	**$$**
	source_path
	BID='xx...xx'
	CODE=A|E|B
	EO=YES|Y|NO|N
	ID=XXXXXXXX
	MULTXMIT=YES|Y|NO|N
	PASSWORD=XXXXXXXX
	TO=YES|Y|NO|N

Command	Parameter
	TRIGGER=YES|Y|NO|N
	XMIT=YES|Y|NO|N

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
$$	indicates that the following data is destined for the mailbox.
source_path	indicates the path name of the source file for the data you want added to the repository.
BID='xx...xx'	identifies the 2–64 byte user batch ID for the batch being added. The entire phrase (BID= 'xx...xx') is required for $$ syntax. You cannot specify a batch number; it is generated internally. To add single-character batch IDs, use standard FTP syntax.
CODE=A|E|B	identifies the formats of the data being added. Three values are possible: A = ASCII, E = EBCDIC, and B = BINARY. The default is ASCII.
EO=YES|Y|NO|N	The EO=Y parameter specifies that this file can only be extracted once and never transmitted. If YES is entered, the batch is marked with a nontransmittable flag. After it is extracted by the host site, it is flagged unextractable. The default is NO.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your site.
MULTXMIT=YES|Y|NO|N	enables or disables multiple transmissions of this batch. This parameter can be abbreviated to MX. **YES|Y—**Indicates that the batch can be transmitted multiple times. If YES is specified, it overrides the **XMIT** parameter and sets it to YES. With MULTXMIT=Y, the added batch is flagged as multitransmittable at the time it is added to the mailbox, but unlike XMIT=Y, the batch is not flagged as transmitted when transmitted successfully. This leaves it eligible for subsequent transmissions that would not be possible if the transmitted flag were set. **NO|N—**Indicates that the batch cannot be transmitted multiple times. This is the default.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.

Parameter	Description
TO=YES|Y|NO|N	enables or disables a *transmit once* capability. **YES|Y**—Specifies that the batch can only be transmitted once. After transmission to the intended remote site, the batch is permanently locked and is flagged as nontransmittable and unextractable. If transmission of a batch with this parameter fails after one or more records have been transmitted, the batch is still locked. To retry the transmission, a new batch must be added from the original source. **NO|N**—Specifies that the batch is not flagged as unextractable. This is the default.
TRIGGER=YES|Y|NO|N	allows files to be rerouted immediately to other remote sites. In order for automatic routing to function, an Auto Connect file must be defined with the **Contact** parameter set to **Forward data to Remote site automatically**. When the characteristics of the batch being added match the selection criteria in this ACD file, the batch is automatically forwarded to the destination specified in the ACD file. See the *Auto connect Definition (ACD)* chapter of the *Connect:Enterprise UNIX Configuration Files Reference Guide* for more information about the ACD file. **YES|Y**—The batch is rerouted if a valid Auto Connect list with matching selection criteria has been defined. **NO|N** —The batch is not forwarded. This is the default.
XMIT=YES|Y|NO|N	determines whether a batch is limited to host site use (in the network where the mailbox exists) or can be distributed to other locations. **YES|Y**—Specifies that the batch is available for transmission to any remote site that identifies the correct mailbox ID. The batch is marked with a requestable flag. With XMIT=Y, the added batch is flagged as transmitted after it is successfully forwarded to another remote site. **NO|N**—Specifies that the batch is available only for host site (Connect:Enterprise) extraction. This parameter indicates that the requestable flag is not set, which restricts remote sites from requesting the batch. This is the default.

$$ put Examples

In the following example, a remote site (ACME) sends a report file to the Connect:Enterprise host site. The data is in ASCII format and is identified with the user batch ID ’sales summary.’ The final destination of the data is the host site, so ACME uses the defaults XMIT=N, MULTXMIT=N, TO=N. Because it does not want to limit host site extraction to a single extract, ACME accepts the default EO=N. The batch is flagged as collected to indicate collection from a remote site. The batch does not have a requestable flag. It can be extracted by a host site **cmuextract** command, but without a requestable flag, it cannot be forwarded to another remote site.

```
ftp> put sales.dat "$$ ID=ACME BID='sales summary'"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=ACME BID='sales summary'.
226 Transfer complete (Batch Number = 105).
45 bytes sent in 0.04915 seconds (0.8941 Kbytes/s)
ftp>
```

Another ASCII file is destined for a single remote site named SMITH after being successfully transmitted to the repository. Use of the XMIT=Y parameter sets the requestable (R) flag on the batch, making it requestable by other remote sites. If the transmission to the other remote site is unsuccessful, the transmitted (T) flag is not set on the batch and it remains available for another attempt. The transmitted (T) flag is set on the batch after it is transmitted successfully and the requestable flag remains set. The transmitted flag makes the batch ineligible for additional transmissions. There is one exception: After the transmitted flag is set, the batch is still available to a remote site that uses a **get** command specifying the batch number.

Even though ACME makes the batch available for forwarding to another remote site, they have allowed host site extraction of the batch with **cmuextract** (TO=N is the default). Extraction at the host site sets the extracted (E) flag for the batch, but this does not affect the availability of the batch for transmission to another remote site, unless the host site **cmuextract** operation deletes the batch by using the DELETE=YES option. This example shows ACME using the ID of SMITH because the remote site SMITH specifies ID=SMITH on the **get** command it issues to collect the available batch. ACME could have used the ID=ACME, but then the SMITH remote site would have had to specify ID=ACME. Either way works. It is up to the two remote sites to decide how they want to use mailbox IDs, considering security administration at the host site (including MAILBOX_LIST designations in the account definition and the content of $*CMUHOME/med/mbxacl.conf*).

```
ftp> put terminate.who "$$ ID=SMITH BID='termination notices' XMIT=Y"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=SMITH BID='termination notices'
XMIT=Y.
226 Transfer complete (Batch Number = 100).
45 bytes sent in 0.001021 seconds (43.04 Kbytes/s)
ftp>
```

ACME forwards a binary file to multiple remote sites through the Connect:Enterprise host site. The batch is flagged as collected and requestable. The batch is not flagged as transmitted upon retransmission from the host site mailbox. The multitransmittable (M) and requestable (R) flags remain in effect after transmissions. The only action a remote site can take to make this batch unavailable for transmission is to logically delete it with a **del** command. Without the deleted (D) or transmitted (T) flags set, a multitransmittable batch remains permanently eligible for transmission.

```
ftp> bin
200 Type set to I.
ftp>  put greetings.doc "$$ ID=ACME BID='holiday greetings' MULTXMIT=Y CODE=B"
200 PORT command successful.
150 Opening BINARY mode data connection for $$ ID=ACME BID='holiday greetings'
MULTXMIT=Y CODE=B.
226 Transfer complete (Batch Number = 99).
41 bytes sent in 0.00102 seconds (39.25 Kbytes/s)
ftp>
```

Note: If the user had not put the session into binary mode by entering the **bin** command, as shown in the preceding example, the **put** command would fail because CODE=B indicates a binary transmission, which requires binary mode to be set. The following example shows this failure, assuming ASCII mode (the default mode when the remote user logs in to the session).

```
ftp> put greetings.doc "$$ ID=ACME BID='holiday greetings' MULTXMIT=Y CODE=B"
200 PORT command successful.
550 Error: Invalid CODE= option specified.
ftp>
```

The next example illustrates that the session must also be in binary mode for EBCDIC transmissions.

To prevent host site extraction yet leave the EBCDIC batch available for a single transmission to the remote site SMITH, ACME can use TO=Y to indicate a *transmit once/transmit only* status for the added batch. The batch displays the collected, requestable, and unextractable flags when the add finishes. Unextractable batches are permanently locked against host site extraction and after they are transmitted to a remote site, the unextractable batch is flagged nontransmittable and transmitted.

Note: If transmission of this batch fails midway, it is still flagged as nontransmittable, but is not flagged transmitted. This result reveals a failed transmission, but the batch cannot be retransmitted. After the nontransmittable flag is set, the batch is permanently locked and the originating station, ACME, must add a duplicate batch to the repository to retry the entire operation.

Like the preceding example, the session illustrated in the next example must be in binary mode because CODE=E (EBCDIC) is specified. Without the **bin** command, as shown in the preceding example, the **put** command fails, as shown in the following example:

```
ftp> put secret.dat "$$ ID=SMITH BID='for your eyes only' TO=Y CODE=E"
200 PORT command successful.
550 Error: Invalid CODE= option specified.
ftp>
```

In the following example, ACME transmits a report file called *sales.dat* to the host site. The data is in EBCDIC format and is identified with the user batch ID 'sales summary'. ACME, the remote site, does not allow the batch to be forwarded to any other remote sites nor extracted at the host site

more than once. The **CODE** parameter must be used because the data format is EBCDIC, not the default (ASCII). This is why the **bin** command is required before the **put** command. Connect:Enterprise flags the batch as collected but not requestable. The EO=Y parameter flags the batch as nontransmittable to reflect the extract only/extract once status of the batch. This batch is permanently locked against transmission and, upon successful extraction, is locked against subsequent extraction with the unextractable flag.

```
ftp> bin
200 Type set to I.
ftp> put sales.dat "$$ ID=ACME BID='sales summary' EO=Y CODE=E"
200 PORT command successful.
150 Opening BINARY mode data connection for $$ ID=ACME BID='sales summary' EO=Y
CODE=E.
226 Transfer complete (Batch Number = 102).
41 bytes sent in 0.001022 seconds (39.18 Kbytes/s)
ftp>
```

The following example allows transmission of a batch to multiple mailboxes in a single command.

```
ftp> put my.text.file "$$ ID=SMITH,ACME,STEVE BATCHID='My multiadd batch'"
200 PORT command successful.
150 Opening ASCII mode data connection for $$ ID=SMITH,ACME,STEVE BATCHID='My
multiadd batch'.
226 Transfer complete (Batch Number = 101,98,99).
45 bytes sent in 0.0247 seconds (1.779 Kbytes/s)
ftp>
```

Invalid put Parameter Combinations

The following combinations of parameters are invalid with the **put** command:

- Do not use XMIT=Y with MULTXMIT=Y.
- Do not use XMIT=Y with TO=Y.
- Do not use XMIT=Y with EO=Y.
- Do not use TO=Y with MULTXMIT=Y.
- Do not use EO=Y with MULTXMIT=Y.
- Do not use EO=Y WITH TO=Y.

Determining Data Format Flag for Inbound Batches

Three factors determine the data format flag for an inbound batch from Connect:Enterprise:

- The data connection mode in use by the FTP client. By default, the FTP client is in ASCII mode. In the absence of the **Format of data received from remote site** parameter specified in the account definition and no CODE= option supplied for the **put** command, the batch is flagged as ASCII. If the remote user issues a **bin** command before a **put** command, the batch is flagged as binary.
- The **Format of data received from remote site** parameter specified in the account definition for the remote site overrides the data connection mode in use by the FTP client.
- The CODE= option supplied on the **put** command overrides both the data connection mode and the **Format of data received from remote site** parameter in the account definition.

FTP Auto Connect issues **bin** and **ascii** commands intelligently before issuing the **put** command. If the **Auto Convert** parameter is set to **ASCII** and is specified in the account definition, the FTP Auto Connect client issues the **ascii** command before the **put** command. If **Auto Convert** is not specified, the data connection mode is determined by the **Outbound batch separation** parameter. If the **Outbound batch separation** parameter is set to **Multiple batches are sent as a single batch**, then the FTP Auto Connect client issues the **ascii** command before the **put** command, if all batches eligible for transmission are ASCII batches. If any of the batches are binary or EBCDIC, FTP Auto Connect **client** issues a **bin** command. If the **Outbound batch separation** parameter is set to **Batches are created as individual files**, then FTP Auto Connect client issues an **ascii** or a **bin** command before issuing the **put** command for every client batch, depending on the data format flag of the batch.

Automatic Routing

Refer to the *System Configuration* chapter of the *Connect:Enterprise UNIX Installation and Administration Guide* for a description of Automatic Routing, which allows batches to be automatically forwarded to remote sites as soon as they are added to a mailbox, for example, with a $$ FTP command.

Remote Site Script Usage

FTP commands and macros can be used as a remote site scripting language for FTP clients. An FTP script can be executed in two different ways:

- Execute a script without reading the initialization file *.netrc*:

```
FTP -n < script.FTP
```

  where *-n* specifies to FTP to start without reading the initialization file.

- Execute a script with initialization by reading the *.netrc* file:

```
FTP hostname < script.FTP
```

Note: Refer to UNIX documentation under FTP commands for further discussion of the *.netrc* file.

If the *netrc* file is not read, the script (*script.FTP)* looks similar to the following example:

```
open fantasia 123
user "bill gandalf"
dir "$$ ID=bill"
binary
get "$$ ID=bill BID=invoices" invoice.dat
put orders.dat "$$ ID=bill BID=orders"
bye
```

The FTP **binary** command is used before the **get** or **put** command. This command specifies to FTP to accept any input without converting it, as if all input or output were binary files.

The **binary** command passes responsibility for converting files (if conversion is necessary) to Connect:Enterprise. The **CONV** parameter in the **get** or **mget** command signals the conversion of an incoming file. The **CODE** parameter in the **put** or **mput** command identifies the format of an outgoing file.

When you instruct FTP to read the *.netrc* file, a sample list of it looks like this:

```
dir "$$"
get "$$" sales.dat
put invent.dat "$$"
bye
```

Notice that this script lacks the FTP **open**, **user**, and **binary** commands. These commands are located in the*.netrc* file with the host name and any other commands for an FTP session.

To redirect all output for a script-driven session from the screen to a file, use the following script, which enables viewing long directory listings, for example.

```
FTP hostname < script.FTP > script.out
```

Chapter 4

SSH SFTP Commands

This chapter describes the following commands available to SSH SFTP remote users using standard SSH SFTP syntax:

- **bye**–Log off of Connect:Enterprise.
- **cd**–Changes the working mailbox ID.
- **get**–Requests a batch of data from the host site.
- **ls**–Requests a formatted listing of batches from the host site.
- **put**–Sends a batch of data to the host site.
- **pwd**–Prints the working mailbox ID.
- **rename**–Renames a batch ID.
- **rm**–Deletes a remote file.

Requirements

Obtain the following information from your host site system administrator prior to executing a Connect:Enterprise SSH SFTP command:

- Mailbox ID
- Password and any additional site-specific security, including information about *mbxacl.conf*, which restricts users from performing specific commands in specific mailbox IDs other than their own.
- IP address or host name and SSH SFTP port
- Both the sending and receiving sites must use SSH SFTP protocol.

Logging On to Connect:Enterprise

If your remote site uses SSH SFTP communications software, use the commands and procedures that follow to access and use Connect:Enterprise:

1. At the command line prompt for your system (shown as $ here), type this command and parameters:

```
$ sftp -o Port=port_number username@host_name
```

   The following table describes the available parameters. All parameters are required.

Parameter	Description
port_number	Optional. The Connect:Enterprise SSH SFTP port listener number. The Connect:Enterprise FTP port listener number. Refer to the Define System functionality of the Site Administration user interface to specify this port.
username	Optional. Default is current login.
host_name	Name of the system running Connect:Enterprise.

2. Depending on the configuration of your client, you are prompted for a passphrase (for public key authentication) or a password (for password authentication). Type the password or passphrase and press **Enter**.

You now have access to Connect:Enterprise. You can send data to and receive data from the repository with the commands described in the following sections.

Logging Off SSH SFTP

When you are finished using the Connect:Enterprise SSH SFTP server, you must log off using the **quit** command. In the following example, the user enters **quit.**

```
sftp> quit
```

cd Command

The **cd** (change directory) command changes the current working mailbox ID to the one specified by the argument. If the mailbox ID does not exist, **cd** warns the user that no batches exist in the mailbox (but allows the user to stay there anyway). The **cd** command only allows users to access mailboxes specified by the **Mailbox list** parameter in their account definition.

The permissions for mailbox IDs are set by the system administrator in the *mbxacl.conf* file.

cd Command Format

You can use only the standard SSH SFTP syntax for the **cd** command, as shown in the following example:

```
sftp> cd /mailbox ID
```

Note: Enter **cd** commands with a forward slash in front of the argument unless you are in the root directory (/).

Standard SSH SFTP cd Syntax

The following table describes the available parameters. Required parameters are in bold.

Parameters	Description
/mailbox_ID	identifies the name of the mailbox ID that becomes the working mailbox ID. This parameter can be up to eight characters. It must be preceded with a forward slash (/). You cannot issue the **cd** command at the root directory.

Standard SSH SFTP cd Examples

Change mailbox from the home mailbox to the user01 mailbox.

```
cd /user01
```

get Command

The **get** command requests one or more files from the repository. Multiple batches with the same batch ID are always concatenated into a single file, no matter what the keyword **Outbound batch separation** is set to in the account definition for the remote site.

The permissions for your mailbox ID are set by the system administrator in the *mbxacl.conf* file. This file indicates mailboxes from which you can retrieve batches.

get Command Format

The following example shows the SSH SFTP syntax for the **get** command:

```
sftp> get remotepath localpath
```

Standard SSH SFTP get Syntax

The following table describes the available parameters. Required parameters are in bold.

remotepath	Mailbox containing files you are requesting followed by a batch ID. If you specify a mailbox as *remotepath*, you will request all batches in the mailbox. If you specify a mailbox and batch ID, you will get only the batch ID you specify. If you specify a mailbox ID and a wildcard, you will get all batches in the mailbox that match the wildcard.
localpath	Specifies the directory on your local computer where you want the requested files to be placed. The default is the current working directory. If *remotepath* is a mailbox, *localpath* must be a directory. If *remotepath* is a mailbox and file name, *localpath* must be a directory and file name. If *remotepath* is a mailbox ID and a wildcard, *localpath* must be a directory.

Standard SSH SFTP get Examples

In the following example, a request is made to get all files in the user01 mailbox and place them in the batches directory on the local computer:

```
get user01 batches
```

In the following example, a request is made to get a batch called batch01 from the user01 mailbox and place it in the batches directory on the local computer and change the name to 01batch:

```
get user01/batch01 batches/01batch
```

In the following example, a request is made to get all batches ending in 01 from the user01 mailbox and place it in the batches directory on the local computer:

```
get user01/*01 batches
```

Unavailable Batches

Batches with the following characteristics are unavailable to remote users. The **get** command does not retrieve batches that are:

- Flagged as nontransmittable (U)
- Flagged as deleted (D)
- Not flagged as requestable (R)
- Flagged as transmitted (T)
- Flagged as incomplete (I)
- Not matching the mailbox ID and/or user batch ID

ls Command

The **ls** command displays a list of batches in a mailbox.

ls Command Format

The following example shows the SSH SFTP syntax for the **ls** command:

```
sftp> ls [mailbox ID][batch ID]
```

Standard SSH SFTP ls Syntax

The following table describes the available parameters:

Parameter	Description
mailbox ID	Identifies the mailbox for which the batches are listed. The default is the current working mailbox.
batch ID	Batch ID or partial name with wildcards to list specific batches.

Standard SSH SFTP ls Examples

The following example lists all batches in the current working mailbox:

```
ls
```

The following example lists all batches in the user01 mailbox:

```
ls user01
```

The following example lists all batches in the user01 that end in 01:

```
ls user01/*01
```

put Command

The **put** command adds one or more batches of data to a mailbox ID. Refer to the Define Accounts or Define Schedules function of the Site Administration user interface Help. for a complete description of **FTP put options**.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file in the *$CMUHOME/med* directory. The contents of this file determine the mailboxes you can add batches to.

put Command Format

The following is the standard SSH SFTP syntax:

```
sftp> put localpath remotepath
```

Standard SSH SFTP put Syntax

The following table describes the available parameters Required parameters are in bold.

Parameter	Description
localpath	Directory containing files you are sending. If you specify a directory as localpath, you will request all files in the directory. If you specify a directory and file name, you will send only the file name you specify. If you specify a directory and a wildcard, you will send all files in the directory that match the wildcard.

Parameter	Description
remotepath	Specifies the mailbox where you want the requested files to be sent. The default is your default mailbox. If *localpath* is a directory, *remotepath* must be a mailbox. If *localpath* is a directory and file name, *remotepath* must be a mailbox and batch ID. If *localpath* is a directory and a wildcard, *remotepath* must be a directory.

Standard SSH SFTP put Examples

In the following example, a request is made to put all files in the batches directory on the local computer and place them in the user01 mailbox:

```
put batches user01
```

In the following example, a request is made to put a batch called 01batch from the batches directory on the local computer and place it in the user01 mailbox and change the name to batch01:

```
put batches/01batch user01/batch01
```

In the following example, a request is made to put all files beginning with 01 from the batches directory on the local computer and place them in the user01 mailbox:

```
put batches/01* user01
```

pwd Command

The **pwd** command displays the current working directory (mailbox ID) at any time during an SSH SFTP session. It prints to standard output, which can be the terminal, unless scripting is being run with standard output set to a file. Different clients have different output.

pwd Command Format

You can only use standard SSH SFTP syntax with the **pwd** command.

Example

This example shows the only way the **pwd** command can be used. The command **pwd** accepts no parameters or arguments. Output will differ, depending on the client you are using.

```
sftp> pwd
Remote working directory: /mailboxID
sftp>
```

rename Command

This command renames a remote batch ID.

rename Command Format

The following example shows the SSH SFTP syntax for the **rename** command:

```
sftp> rename [/mailboxID/]oldbatchID [/mailboxID/]newbatchID
```

Standard SSH SFTP rename Syntax

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
oldbatchID	Identifies the batch to change.
newbatchID	New batch ID.
mailboxID	Identifies the mailbox location of the batch.

Standard SSH SFTP rename Examples

The following example renames a batch in the user01 mailbox from batch01 to 01batch:

```
rename /user01/batch01 /user01/01batch
```

The following example renames a batch in the current working mailbox from batch01 to 01batch:

```
rename batch01 01batch
```

rm Command

The **rm** command deletes the specified batch ID.

rm Command Format

The following example shows the SSH SFTP syntax for the **rm** command:

```
sftp> rm [/mailbox ID/] batch ID
```

Standard SSH SFTP rm Syntax

The following tabled describes the available parameters. Required parameters are in bold.

Parameter	Description
batch ID	Batch to delete.
mailbox ID	Mailbox ID where the batch ID is located. Default is the current mailbox.

Standard SSH SFTP rm Examples

The following example deletes batch 01 from the current working mailbox:

```
sftp> rm batch01
```

The following example deletes batch 01 from the user01 mailbox:

```
sftp> rm /user01/batch01
```

Chapter 5

SSH SFTP $$ Commands

Connect:Enterprise accepts commands in two formats: Mailbox ($$) syntax and standard SSH SFTP syntax.

This chapter describes the commands available to SSH SFTP remote users using $$ syntax. Refer to Chapter 4, *SSH SFTP Commands*, for information about standard SSH SFTP commands.

- **rm**—Flags a batch of data as deleted.
- **ls**—Requests a formatted listing of batches from the host site.
- **get**—Requests a batch of data from the host site.
- **put**—Sends a batch of data to the host site.

Logging Off SSH SFTP

When you are finished using the Connect:Enterprise SSH SFTP server, you must log off using one of three SSH SFTP commands: **bye**, **exit** or **quit**. In the following example, the user types **quit**.

```
sftp> quit
```

rm Command

This command flags a specific batch of data as logically deleted from your mailbox, by setting the deleted (D) flag.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file. The contents of this file determine the mailboxes you can delete batches from. If you enter a **rm** command for a mailbox you do not have access to, the message *Permission denied* is returned.

When a batch is logically deleted, the D flag is set for that batch. The host site operator can restore the batch using the **cmustatus** command or physically erase it using the **cmuerase** command.

rm Command Format

The following example shows the $$ syntax for the **rm** command:

```
sftp> rm "$$ parameters"
```

$$ rm Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both kinds of parameters are listed alphabetically.

With the $$ syntax, you can use any number of blanks between parameters, but blanks are not allowed within parameters, except for the batch ID. All text following the **rm** command (including $$) must be enclosed in one set of double quotation marks.

Command	Parameter
rm	**$$**
	BID='xx...xx'|#nnnnnnnn
	ID=XXXXXXXX

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
$$	indicates that mailbox syntax is being used.
BID='xx..xx'| #nnnnnnnn	indicates either the user batch ID or the batch number. The user batch ID can be 1–64 characters. Wildcard specifications (like an asterisk, *) are supported. A single-word batch ID does not require quotation marks. To specify a multiword BID with spaces, enclose the BID in single quotation marks (' '). To specify a batch number, precede the batch number with a pound sign (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign, and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.
ID=xxxxxxxx	indicates the valid mailbox ID of the site requesting the deletion and can be 1–8 characters. Wildcard specifications (like an asterisk, *) are supported.

$$ rm Examples

In the following example, ACME deletes a specific batch by specifying its batch number. This command logically deletes batch number 775 and sets the deleted (D) flag. Deleted batches are not available for transmission, even if the requestable flag is on, and a specific batch number is specified on the request. Deleted batches can be extracted at the host site, if the specific batch number is specified with the **cmuextract** command. Batches can be restored only at the host site with the **cmustatus** command.

```
sftp> rm "$$ ID=ACME BID=#775"
```

In the next example, ACME narrows the deletion of ACME batches to those that have the user batch ID 'invoices' as well as the mailbox ID ACME.

```
sftp> rm "$$ ID=ACME BID='invoices'"
```

ls Command

The **ls** command displays information about batches in the current working mailbox.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file. These permissions determine the mailboxes you can obtain directory listings for. If you enter an **ls** command for a mailbox you do not have access to, you get the message *Permission denied*.

Batches can be selected (filtered) by mailbox ID alone, mailbox ID and batch ID, by batch number, originator, flags, and start and end times.

The contents are displayed either to standard output or to a local file if one is specified as an argument.

ls Command Format

The following shows the $$ syntax for the **ls** command:

```
sftp> ls "$$ parameters"
```

$$ ls Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both kinds of parameters are listed alphabetically.

You can use any number of blanks between parameters, but not within parameters. All text following the **ls** command (including $$) must be enclosed in one set of double quotation marks

Command	Parameter
ls/dir	**$$**
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ORIG=XXXXXXXX
	PASSWORD=XXXXXXXX
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
$$	indicates that mailbox syntax is being used.
FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I deletes all batches that were added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I deletes the batches that are neither added locally nor incomplete. The syntax CM!I deletes batches that are collected remotely, are multitransmittable, and not incomplete.

Parameter	Description
FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for deletion. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **ls** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **ls** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 deletes all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. FTIME=020101:1315 limits the **ls** command to those batches created on or after January 1, 2002 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ID=xxxxxxxx	indicates the valid mailbox ID of the site requesting the deletion and can be 1–8 characters. Wildcard specifications (like an asterisk, *) are supported.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.

Parameter	Description
TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on which or before which a batch must have been created to be eligible for deletion. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **ls** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **ls** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 TTIME=20020101 deletes all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **ls** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

Output

After an **ls** command is issued, Connect:Enterprise scans the repository for the designated batches and returns that information to the remote site in the following format:

```
-CR--M----SSH B user01          18      105 Feb 10 16:25 batch01
-CR--M----SSH B user01         282      613 Feb 27 16:53 batch02
-CR--M----SSH B user02         286      197 Feb 27 17:12 batch03
-C--------SSH B user02         422    16113 Mar 04 2003  batch04
-C--------SSH B user02         423    14057 Mar 04 2003  batch05
```

The following table describes the output:

Parameter	Description
flags	One or more flags identifying the current status of a batch in the repository. Refer to *Flags* on page 11 for definitions of these flags.
protocol	indicates SSH as the transmission protocol used to place the batch into the specified mailbox.
format	indicates the data format of the batch. Valid values are ASC (ASCII), BIN (binary), or EBC (EBCDIC).
originator	indicates the mailbox ID of the originator.
nnnnnnnn	The 8-digit batch number assigned to the batch.
bbbbbbbbbb	The total byte count of the batch.
YYMMDD–HHMM	The date and time that the batch was added to the repository.
	The first 24 characters of the batch's user batch ID. If the user batch ID is longer than 24 characters, it is shortened to the first 23 characters of the literal with the 24th character becoming a plus sign (+).

$$ Syntax ls Examples

In the following example, ACME generates a listing of all batches stored in the repository whose mailbox ID is ACME. With no destination file name specified, the batch information records are displayed on screen. If ACME is the remote site entering these commands, then to obtain batches in the repository called ACME, specifying ID=ACME is optional.

```
sftp> ls "$$ ID=ACME"
-CR--M----SSH B ACME          18      105 Feb 10 16:25 batch01
-CR--M----SSH B ACME         282      613 Feb 27 16:53 batch02
-CR--M----SSH B ACME         286      197 Feb 27 17:12 batch03
-C--------SSH B ACME         422    16113 Mar 04 2003  batch04
-C--------SSH B ACME         423    14057 Mar 04 2003  batch05
Total Number of batches listed: 5
sftp>
```

In the next example, ACME uses the default for the ID and PASSWORD parameters, that is, those specified when the user command was issued to log on. Batch information records are returned for all mailbox batches with a mailbox ID matching the login ID. In this case, the remote user is *steve*, so the output shown is for the mailbox with the ID *steve*.

```
sftp> ls $$
-CR--M----SSH B steve          28      105 Apr 02 14:01 batch01
-CR--M----SSH B steve         334      613 Apr 10 14:33 batch02
-CR--M----SSH B steve         368      197 Apr 11 15:18 batch03
-C--------SSH B steve         402    16113 May 12 2003 batch04
-C--------SSH B steve         511    14057 May 14 2003 batch05
Total Number of batches listed: 5
sftp>
```

The following example shows the use of the ORIG option. The command returns all batches whose originator ID is the *originator name* indicated (within the constraints set forth by the system administrator in the *mbxacl.conf* file). In other words, if the current directory has no batches available to the current user (based on that user's permissions), then the **ls** command does not return any batches.

Note the following three batches (batch numbers 20, 14 and 13). These are a subset of the batches shown in the first example. In that example, the output included two batches with the batch ID *test batch* and two more with the batch ID *orders*. This indicates to the user that the two other batches (batch numbers 19 and 4) were not originated by *steve*.

```
sftp> ls "$$ ID=ACME ORIG=steve"
-CR--M----SSH B ACME          18      105 Feb 10 16:25 batch01
-CR--M----SSH B ACME         282      613 Feb 27 16:53 batch02
-CR--M----SSH B ACME         286      197 Feb 27 17:12 batch03
Total Number of batches listed: 3
226 Transfer complete.
sftp>
```

The following example give a directory listing of all batches originated by *dave* at or after September 29, 2002, 1:25 pm.

```
sftp> ls "$$ ORIG=dave FTIME=20040131:1325"
-CR--M----SSH  B ACME         22      106 Feb 01 16:25 batch01
-CR--M----SSH  B ACME        131      708 Feb 02 16:53 batch02
-CR--M----SSH  D ACME        225      181 Feb 23 17:12 batch03
Total Number of batches listed: 3
sftp>
```

The **Flags** parameter is a filtering identifier. The next example is identical to the preceding one, except for the Flags=!D. Note that unlike the preceding example, batch 225 (with the D flag set) is not listed.

```
sftp> ls "$$ ORIG=dave FTIME=20020929:1325 FLAGS=!D"
-CR--M----SSH  A ACME         22      106 Oct 10 16:25 batch01
-CR--M----SSH  A ACME        131      708 Oct 27 16:53 batch02
Total Number of batches listed: 2
sftp>
```

The following example returns all batches that are flagged for multiple transmission (M) and are requestable (R).

```
sftp> ls "$$ FLAGS=MR"
```

get Command

The **get** command requests a single file or multiple files from the repository. This single file contains all the batches matching the selection criteria. For example, if ID=j* is specified, all batches contained in mailboxes with names beginning with j are sent as one file.

The **get** command copies a file from the repository to the local directory. Multiple batches with the same batch ID are always concatenated into a single file, no matter what BCHSEP is set to.

The permissions for your mailbox are set by the system administrator in the *mbxacl.conf* file. This file indicates where you can retrieve batches from.

Failing to supply the required output file name when issuing a **get** command can create a file in the current directory with a name equal to the string contained with the double quotation marks portion of the **get** command. This file contains the data you intended to obtain from the Connect:Enterprise host. However, the unusual file name can cause problems when you attempt to view, copy, move, or even remove the file. For example, when you issue the following command:

```
get "$$ ID=dave"
```

After ending your SSH SFTP session, you find a file in the current directory named:

```
$$ ID=dave
```

The following example shows the situation described in the preceding example. During the SSH SFTP session, you enter the **get** command. Then you issue the **quit** command to exit the session and return to the shell prompt. Next, the **ls -al** command displays files in the local directory, including a file called *$$ ID=dave*.

```
sftp> get "$$ ID=dave"
sftp> quit
mynode> ls -al
-rw-r--r--   1 steve    develop      221 Sep 29 13:43 $$ ID=dave
drwxr-xr-x   2 steve    develop      512 Sep 29 13:43 ./
drwxr-xr-x  17 steve    develop      512 Aug 27 16:11 ../
-rw-r--r--   1 steve    develop       72 Sep 25 16:09 ACME
-rw-r--r--   1 steve    develop       69 Sep 29 13:27 dave
-rw-r--r--   1 steve    develop     5949 Oct  9 1996  sample_rsd
-rw-r--r--   1 steve    develop      152 Sep 25 14:39 steve
mynode>
```

The $$ and space characters can cause problems. They must be prefixed with the back slash (\) when the file name is referenced on the command line. The following example renames the file to *output.dat*.

```
mv   $$\ ID=dave  output.dat
```

A space follows the third back slash (\) in the example.

get Command Format

The following example shows the $$ syntax for the **get** command:

```
sftp> get "$$ parameters" [dest_filename]
```

$$ get Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both types of parameters are listed alphabetically.

The keyword is followed by double quotation marks, two dollar signs, any parameters needed, and an ending double quotation marks. The dest_filename must be the last option after the final double quotation marks. The double quotation marks are required. Refer to the syntax example in the *get Command Format* section on *get Command Format* on page 99.

You can use any number of blanks between parameters, but not within parameters, except for the BID, which must be enclosed in single quotes if spaces are embedded.

Command	Parameter
get	**$$**
	BID='xx...xx'|#nnnnnnnn
	dest_filename
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ONEBATCH=YES|Y|NO|N
	ORIG=XXXXXXXX
	RECEIPT
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
$$	indicates that the following data is destined for the mailbox.
BID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. It can be 1–64 characters. To create a multiword BID, enclose the user batch ID in single quotation marks (' '). Single-word BIDs (with no spaces) do not require quotes. Wildcard specifications (like an asterisk, *) are supported. To specify a batch number, use a pound sign followed by the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign. If neither the **BID** nor the **ID** parameter is specified, all batches in the current mailbox are selected and concatenated into one file.

Parameter	Description
dest_filename	identifies the destination file name and directory path name for the data you want from the mailbox. Depending on the file system being used, the **get** command can truncate file names (for example, to the 8.3 format on DOS). Therefore, always supply the dest_filename argument if it is known.
FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I specifies to get all batches added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I specifies a **get** request the batches that are neither added locally nor incomplete. The syntax CM!I specifies a **get** request for batches that were collected remotely, are multitransmittable, and are not incomplete.
FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for retrieval. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **get** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **get** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. Specifying FTIME=010101:1315 limits the **get** command to batches created on or after January 1, 2002 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm Specifies on or after the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID of the remote site requesting the data. Wildcard specifications (like an asterisk, *) are supported. If neither the **BID** nor the **ID** parameter is specified, all batches in the current mailbox are selected and concatenated into one file.

Parameter	Description
ONEBATCH=YES|Y|NO|N	indicates that only the first batch matching the selection criteria is to be transmitted to the remote site in response to the **get** command. **YES|Y** —Specifies to transmit only a single batch to the remote site. This is especially useful with the **mget** command: when a specific batch number has not been specified and you want to prevent concatenation of multiple batches by limiting the transmission to one batch, use this parameter. **NO|N** —Specifies to transmit all batches that conform to the request to the remote site. This is the default.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.
RECEIPT	sends the log batch called <<ACTIVITY LOG>>. It is a log of all the ADD/REQ activity performed by the remote user, and is created if the account parameter **Log receipt of batch** is checked. This batch can be used as a receipt by the remote user. This batch is reset by the command **del "$$receipt"**. This parameter cannot be used with any other optional parameter.
TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on or before which a batch must have been created to be eligible for retrieval. If the **FTIME** parameter is omitted, the creation date and time of oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **get** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **get** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20000101 lists all batches created on or between December 31, 1999 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. The entry TTIME=011231:1400 limits the **get** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn Specifies—on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

$$ get Examples

In the first example, ACME retrieves all the eligible batches that have been added to the repository at the host site. Eligible batches are batches that have a requestable flag but do not yet have a transmitted (T), nontransmittable (N), or deleted (D) flag. ACME can view these status flags by first issuing a **ls** command to receive a listing of the ACME batches. Having taken the default parameter, CONV=N, no transmission code conversions are done. Batches of all types (ASCII, EBCDIC, and binary) are retrieved.

This SSH SFTP **get** command collects one or more eligible batches as a single concatenated file named *receive.dat.* Because the default batch separation method BCHSEP=NO is in effect, as each batch in the concatenation is successfully transmitted, Connect:Enterprise immediately flags it as transmitted. Although the requestable flags are retained after transmission, if ACME issues this same **get** command in the future, those batches are no longer eligible. However, newly added batches, lacking transmitted flags, are eligible with one exception: If the batch is originally added to the repository with a MULTXMIT=Y parameter, it has a multitransmittable flag that prevents the batch from ever being flagged as transmitted. This leaves it eligible for transmission to other sites or to the same site multiple times.

```
sftp> get "$$ ID=ACME" receive.dat
sftp>
```

ACME can use the **get** command shown in the preceding example to request a specific batch, having previously obtained batch numbers from an **ls** report. This command can be used to force retransmission of a previously transmitted batch or to select a unique batch. This command does not permit collection of a deleted batch nor a nontransmittable batch. It does not permit collection of a batch that does not have a requestable (R) flag.

In the following example, ACME requests the <<ACTIVITY LOG>> batch maintained by Connect:Enterprise for ACME. This batch contains the log of all ADD and REQ operations performed by ACME. This batch is maintained only if Connect:Enterprise has enabled the verification of data delivery feature for the remote site by checking the **Create log batch?** parameter in the ACME account definition.

To prevent confusion with other batches, explicitly request a receipt by indicating:

```
sftp> get "$$receipt"
sftp>
```

The next example is similar to the preceding example, but the <<ACTIVITY LOG>> batch is extracted to a file called *xxx*.

```
sftp> get "$$receipt" xxx
sftp>
```

The following command returns all batches created between the FTIME and the TTIME time stamps and concatenates them into one file called *local filename*.

```
sftp> get "$$ FTIME=19990929:1300 TTIME=990929:1500" "local filename"
sftp>
```

If you do not enclose *local filename* in double quotes, then the file created on the remote system is called *local* instead of *local filename*.

Unavailable Batches

Batches are unavailable in the following circumstances:

- Batches flagged as nontransmittable are not available to Remote Connect **$$REQUEST** or **get** commands.
- Batches flagged as deleted (D) are not available to Remote Connect **$$REQUEST** or **get** commands.
- Batches lacking the requestable (R) flag are not available to Remote Connect **$$REQUEST** or **get** commands. These were collected from a remote site that did not use XMIT=Y, MULTXMIT=Y, or TO=Y on the **$$ADD** card and thus, are not eligible for retransmission.
- Batches flagged as transmitted (T) are not available to Remote Connect **$$REQUEST** or **get** commands unless a batch number is indicated.
- Batches flagged as incomplete are not available to Remote Connect **$$REQUEST** or **get** commands unless a batch number is indicated.
- Batches that do not match the specified mailbox ID and/or user batch ID are not available.

put Command

The **put** command adds a single batch of data to the current working mailbox. The user-specified parameters control characteristics of the batch being sent.

The permissions are set for your mailbox by the system administrator in the *mbxacl.conf* file. The contents of this file determine the mailboxes to which you can add batches with the **put** command.

put Command Format

The following example shows the $$ syntax for the **put** command:

```
sftp> put source_path "$$ parameters"
```

$$ put Syntax

Required parameters are in bold font in the following table. Optional parameters are in plain font.

The **source_path** parameter must be the first option, immediately after the **put** command. This is followed by a double quotation mark, two dollar signs, any parameters needed and an ending double quotation mark. The double quotation marks are required.

Command	Parameter
put	**$$**
	source_path
	BID=’xx...xx’
	EO=YES|Y|NO|N
	ID=XXXXXXXX
	MULTXMIT=YES|Y|NO|N
	PASSWORD=XXXXXXXX
	TO=YES|Y|NO|N
	TRIGGER=YES|Y|NO|N
	XMIT=YES|Y|NO|N

The following table describes the available parameters. Required parameters are in bold. Parameters are case-sensitive.

Parameter	Description
$$	indicates that the preceding data is destined for the mailbox.
source_path	indicates the path name of the source file for the data you want added to the repository.
BID=’xx...xx’	identifies the 2–64 byte user batch ID for the batch being added. The entire phrase (BID= ’xx...xx’) is required for $$ syntax. You cannot specify a batch number; it is generated internally. To add single-character batch IDs, use standard FTP syntax.
EO=YES|Y|NO|N	The EO=Y parameter specifies that this file can only be extracted once and never transmitted. If YES is entered, the batch is marked with a nontransmittable flag. After it is extracted by the host site, it is flagged unextractable. The default is NO.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your site.
MULTXMIT=YES|Y|NO|N	enables or disables multiple transmissions of this batch. This parameter can be abbreviated to MX. **YES|Y** —Indicates that the batch can be transmitted multiple times. If YES is specified, it overrides the **XMIT** parameter and sets it to YES. With MULTXMIT=Y, the added batch is flagged as multitransmittable at the time it is added to the mailbox, but unlike XMIT=Y, the batch is not flagged as transmitted when transmitted successfully. This leaves it eligible for subsequent transmissions that would not be possible if the transmitted flag were set. **NO|N** —Indicates that the batch cannot be transmitted multiple times. This is the default.

Parameter	Description
PASSWORD=xxxx xxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.
TO=YES|Y|NO|N	enables or disables a *transmit once* capability. **YES|Y**—Specifies that the batch can only be transmitted once. After transmission to the intended remote site, the batch is permanently locked and is flagged as nontransmittable and unextractable. If transmission of a batch with this parameter fails after one or more records have been transmitted, the batch is still locked. To retry the transmission, a new batch must be added from the original source. **NO|N**—Specifies that the batch is not flagged as unextractable. This is the default.
TRIGGER=YES|Y|NO|N	allows files to be rerouted immediately to other remote sites. In order for automatic routing to function, an Auto Connect file must be defined with the **Contact** parameter set to **Forward data to Remote site automatically**. When the characteristics of the batch being added match the selection criteria in this ACD file, the batch is automatically forwarded to the destination specified in the ACD file. See the *Auto connect Definition (ACD)* chapter of the *Connect:Enterprise for UNIX Configuration Files Reference Guide* for more information about the ACD file. **YES|Y**—The batch is rerouted if a valid Auto Connect list with matching selection criteria has been defined. **NO|N**—The batch is not forwarded. This is the default.
XMIT=YES|Y|NO|N	determines whether a batch is limited to host site use (in the network where the mailbox exists) or can be distributed to other locations. **YES|Y**—Specifies that the batch is available for transmission to any remote site that identifies the correct mailbox ID. The batch is marked with a requestable flag. With XMIT=Y, the added batch is flagged as transmitted after it is successfully forwarded to another remote site. **NO|N**—Specifies that the batch is available only for host site (Connect:Enterprise) extraction. This parameter indicates that the requestable flag is not set, which restricts remote sites from requesting the batch. This is the default.

$$ put Examples

In the following example, a remote site (ACME) sends a report file to the Connect:Enterprise host site. The data is in ASCII format and is identified with the user batch ID 'sales summary.' The final destination of the data is the host site, so ACME uses the defaults XMIT=N, MULTXMIT=N, TO=N. Because it does not want to limit host site extraction to a single extract, ACME accepts the default EO=N. The batch is flagged as collected to indicate collection from a remote site. The batch does not have a requestable flag. It can be extracted by a host site **cmuextract** command, but without a requestable flag, it cannot be forwarded to another remote site.

```
sftp> put sales.dat "$$ ID=ACME BID='sales summary'"
sftp>
```

Another ASCII file is destined for a single remote site named SMITH after being successfully transmitted to the repository. Use of the XMIT=Y parameter sets the requestable (R) flag on the batch, making it requestable by other remote sites. If the transmission to the other remote site is

unsuccessful, the transmitted (T) flag is not set on the batch and it remains available for another attempt. The transmitted (T) flag is set on the batch after it is transmitted successfully and the requestable flag remains set. The transmitted flag makes the batch ineligible for additional transmissions. There is one exception: After the transmitted flag is set, the batch is still available to a remote site that uses a **get** command specifying the batch number.

Even though ACME makes the batch available for forwarding to another remote site, they have allowed host site extraction of the batch with **cmuextract** (TO=N is the default). Extraction at the host site sets the extracted (E) flag for the batch, but this does not affect the availability of the batch for transmission to another remote site, unless the host site **cmuextract** operation deletes the batch by using the DELETE=YES option. This example shows ACME using the ID of SMITH because the remote site SMITH specifies ID=SMITH on the **get** command it issues to collect the available batch. ACME could have used the ID=ACME, but then the SMITH remote site would have had to specify ID=ACME. Either way works. It is up to the two remote sites to decide how they want to use mailbox IDs, considering security administration at the host site (including MAILBOX_LIST designations in the account definitions and the content of $*CMUHOME/med/mbxacl.conf*).

```
sftp> put terminate.who "$$ ID=SMITH BID='termination notices' XMIT=Y"
sftp>
```

ACME forwards a binary file to multiple remote sites through the Connect:Enterprise host site. The batch is flagged as collected and requestable. The batch is not flagged as transmitted upon retransmission from the host site mailbox. The multitransmittable (M) and requestable (R) flags remain in effect after transmissions. The only action a remote site can take to make this batch unavailable for transmission is to logically delete it with a **del** command. Without the deleted (D) or transmitted (T) flags set, a multitransmittable batch remains permanently eligible for transmission.

```
sftp> put greetings.doc "$$ID=ACME BID='holiday greetings' MULTXMIT=Y CODE=B"
sftp>
```

The following example allows transmission of a batch to multiple mailboxes in a single command.

```
sftp> put my.text.file "$$ ID=SMITH,ACME,STEVE BATCHID='My multiadd batch'"
sftp>
```

Invalid put Parameter Combinations

The following combinations of parameters are invalid with the **put** command:

- Do not use XMIT=Y with MULTXMIT=Y.
- Do not use XMIT=Y with TO=Y.
- Do not use XMIT=Y with EO=Y.
- Do not use TO=Y with MULTXMIT=Y.
- Do not use EO=Y with MULTXMIT=Y.
- Do not use EO=Y WITH TO=Y.

Automatic Routing

Refer to the *System Configuration* chapter of the *Connect:Enterprise UNIX Installation and Administration Guide* for a description of Automatic Routing, which allows batches to be automatically forwarded to remote sites as soon as they are added to a mailbox, for example, with a $$ SSH SFTP command.

Chapter 6

The SSH SCP Command

Secure Copy (SCP) is a subprotocol of SSH-2. SCP uses secure shell encryption to protect data transferred across a network or over the Internet. You can use any SCP client with Connect:Enterprise.

Adding a File with Standard Syntax

You can add files to Connect:Enterprise using SCP through a command line.

Following is the basic format for the SCP command for adding. There are more parameters available to SCP that alter the behavior of you SCP client. The parameters included in this section are relevant to exchanging files with Connect:Enterprise. For information on other SCP parameters, refer to the **scp --help** information.

```
scp -P port -C file1 [[user@]cehost:/mailboxID/]batchID
```

The following table describes the command elements. Required elements are in bold:

Element	Description
-P *port*	Indicates the SSH port to connect to. This is relevant if you are sending data to Connect:Enterprise and need to specify a specific port other than the client's default port.
-C	If you are sending data to Connect:Enterprise, this parameter indicates that the incoming data is compressed. If you are requesting data from Connect:Enterprise, this parameter instructs Connect:Enterprise to compress the data before sending.

Element	Description
-c *cipher*	If you want to encrypt the data you are sending, this parameter specifies the cipher to use. Connect:Enterprise supports: ◆ aes128-cbc ◆ 3des-cbc ◆ blowfish-cbc ◆ cast128-cbc ◆ arcfour ◆ aes192-cbc ◆ aes256-cbc
file1	Indicates the path and file name that you are sending.
[user@]cehost:/mailboxID/]batchID	user = destination mailbox cehost = the host where Connect:Enterprise is installed mailboxID = destination mailbox batch ID = the batch name for the file you are adding.

Add Examples

The following example adds file01 from the current directory on the local computer to the user01 mailbox through port 9999 on cehost.

```
$ scp -P 9999 file01 user01@cehost:
```

The following example compresses the file then adds file01 from the current directory on the local computer to the user01 mailbox through port 9999 on cehost.

```
$ scp -C -P 9999 file01 user01@cehost:
```

The following example compresses the file then add file01 from the current directory on the local computer to the user01 mailbox through port 9999 on cehost and changes the name of the file to batch01.

```
$ scp -C -P 9999 file01 user01@cehost:batch01
```

The following example compresses the file and encrypts the file using aes128-cbc then adds file01 from the current directory on the local computer to the user01 mailbox through port 9999 on cehost and changes the name of the file to batch01.

```
$ scp -C -c aes123-cbc -P 9999 file01 user01@cehost:batch01
```

Extracting a File with Standard Syntax

You can extract files from Connect:Enterprise using SCP through a command line.

Following is the basic format for the SCP command for extracting. There are more parameters available to SCP that alter the behavior of you SCP client. The parameters included in this section are relevant to exchanging files with Connect:Enterprise. For information on other SCP parameters, refer to the **scp --help** information.

```
scp -C -P port [user@]host2:/mailboxID/file1 file2
```

The following table describes the command elements. Required elements are in bold:

Element	Description
-P *port*	Indicates the SSH port to connect to. This is relevant if you are sending data to Connect:Enterprise and need to specify a specific port other than the client's default port.
[user@]cehost:/mailboxID/batchID	Indicates the source host where Connect:Enterprise is installed, the mailbox, and batchID. Because of the platform dependencies, extracting batches using wildcards to identify Mailbox ID or Batch ID with standard-syntax SCP commands can produce uncertain results. Wildcards should be used in scp only with $$-syntax commands to request batches.
file2	Indicates the path and file name that you are sending.
-C	If you are sending data to Connect:Enterprise, this parameter indicates that the incoming data is compressed. If you are requesting data from Connect:Enterprise, this parameter instructs Connect:Enterprise to compress the data before sending.
-p	Instructs SCP to maintain the timestamp information of the original file.

Extract Examples

The following example extracts batch01 from the user01 mailbox through port 9999 on cehost to the current directory on the local computer and changes the name to file01:

```
$ scp -P 9999 user01@cehost:/user01/batch01 file01
```

The following example extracts batch01 from the user01 mailbox through port 9999 on cehost to the current directory on the local computer and changes the name to file01 and compresses the batch during transfer:

```
$ scp -P 9999 -C user01@cehost:/user01/batch01 file01
```

The following example extracts batch01 from the user01 mailbox through port 9999 on cehost to the current directory on the local computer and retains the original timestamp of the batch:

```
$ scp -P 9999 -p user01@cehost:/user01/batch01 file01
```

Adding a file with $$ Syntax

In addition to standard SCP syntax, you can use $$ syntax with an SCP client. Use the following format to add a file:

```
scp -P port localfilename 'ceuhost:\$\$keyword=value\ keyword=value'
```

The following table describes the command elements. Required elements are in bold:

Element	Description
-P *port*	Indicates the SSH port to connect to. This is relevant if you are sending data to Connect:Enterprise and need to specify a specific port other than the client's default port.
localfilename	Indicates the file name that you are sending.
-C	If you are sending data to Connect:Enterprise, this parameter indicates that the incoming data is compressed. If you are requesting data from Connect:Enterprise, this parameter instructs Connect:Enterprise to compress the data before sending.
-p	Instructs SCP to maintain the timestamp information of the original file.
ceuhost	The name of the Connect:Enterprise server you are connecting to.
\$\$	Indicates that the keywords that follow are parameters specific to Connect:Enterprise.
keyword	A keyword that represents a Connect:Enterprise parameter. Refer to the *Available Keywords* on page 112.
value	The value associated with the keyword. Refer to the *Available Keywords* on page 112.

Available Keywords

The following table describes the available Connect:Enterprise parameters. Required parameters are in bold. These parameters are not case-sensitive.

Parameter	Description
BID='xx...xx'	identifies the 2–64 byte user batch ID for the batch being added. The entire phrase (BID= 'xx...xx') is required for $$ syntax. You cannot specify a batch number; it is generated internally. To add single-character batch IDs, use standard SCP syntax.

Parameter	Description
EO=YES|Y|NO|N	The EO=Y parameter specifies that this file can only be extracted once and never transmitted. If YES is entered, the batch is marked with a nontransmittable flag. After it is extracted by the host site, it is flagged unextractable. The default is NO.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your site. If you are adding multiple files to a mailbox using SCP, you cannot use this parameter.
MULTXMIT=YES|Y|NO|N	enables or disables multiple transmissions of this batch. This parameter can be abbreviated to MX. **YES|Y—**Indicates that the batch can be transmitted multiple times. If YES is specified, it overrides the **XMIT** parameter and sets it to YES. With MULTXMIT=Y, the added batch is flagged as multitransmittable at the time it is added to the mailbox, but unlike XMIT=Y, the batch is not flagged as transmitted when transmitted successfully. This leaves it eligible for subsequent transmissions that would not be possible if the transmitted flag were set. **NO|N—**Indicates that the batch cannot be transmitted multiple times. This is the default.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.
TO=YES|Y|NO|N	enables or disables a *transmit once* capability. **YES|Y—**Specifies that the batch can only be transmitted once. After transmission to the intended remote site, the batch is permanently locked and is flagged as nontransmittable and unextractable. If transmission of a batch with this parameter fails after one or more records have been transmitted, the batch is still locked. To retry the transmission, a new batch must be added from the original source. **NO|N—**Specifies that the batch is not flagged as unextractable. This is the default.
TRIGGER=YES|Y|NO|N	allows files to be rerouted immediately to other remote sites. In order for automatic routing to function, an Auto Connect file must be defined with the **Contact** parameter set to **Forward data to Remote site automatically**. When the characteristics of the batch being added match the selection criteria in this ACD file, the batch is automatically forwarded to the destination specified in the ACD file. See the *Auto connect Definition (ACD)* chapter of the *Connect:Enterprise UNIX Configuration Files Reference Guide* for more information about the ACD file. **YES|Y—**The batch is rerouted if a valid Auto Connect list with matching selection criteria has been defined. **NO|N—**The batch is not forwarded. This is the default.

Parameter	Description
XMIT=YES|Y|NO|N	determines whether a batch is limited to host site use (in the network where the mailbox exists) or can be distributed to other locations. **YES|Y** – Specifies that the batch is available for transmission to any remote site that identifies the correct mailbox ID. The batch is marked with a requestable flag. With XMIT=Y, the added batch is flagged as transmitted after it is successfully forwarded to another remote site. **NO|N** – Specifies that the batch is available only for host site (Connect:Enterprise) extraction. This parameter indicates that the requestable flag is not set, which restricts remote sites from requesting the batch. This is the default.

Add Examples Using $$

The following example adds file01 from the current directory on the local computer through port 9999 on cehost and uses $$ keywords to specify the user02 mailbox and change the name to file name batch01.

```
$ scp -P 9999 file01 'user01@cehost:\$\$id=user02\ bid=batch01'
```

The following example adds file01 from the current directory on the local computer through port 9999 on cehost and uses $$ keywords to specify the user02 mailbox, change the file name to batch01, and add the T flag (transmit) flag to the batch.

```
$ scp -P 9999 file01 'user01@cehost:\$\$id=user02\ bid=batch01\ xmit=y'
```

Extracting a file with $$ Syntax

In addition to standard SCP syntax, you can use $$ syntax with an SCP client. Use the following format to add a file:

```
scp -P port 'ceuhost:\$\$keyword=value\ keyword=value' localfilename
```

The following table describes the command elements:

Element	Description
-P *port*	Indicates the SSH port to connect to. This is relevant if you are sending data to Connect:Enterprise and need to specify a specific port other than the client's default port.
-C	If you are sending data to Connect:Enterprise, this parameter indicates that the incoming data is compressed. If you are requesting data from Connect:Enterprise, this parameter instructs Connect:Enterprise to compress the data before sending.

Element	Description
ceuhost	The name of the Connect:Enterprise server you are connecting to.
\$\$	Indicates that the parameters that follow are parameters specific to Connect:Enterprise.
keyword	A keyword that represents a Connect:Enterprise parameter. Refer to the *Available Keywords* on page 115.
value	The value associated with the keyword. Refer to the *Available Keywords* on page 115.
localfilename	Indicates what to name the file on your local system.

Available Keywords

The following table describes the available Connect:Enterprise parameters. Required parameters are in bold. These parameters are not case-sensitive.

Parameter	Description

BID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. It can be 1–64 characters.
	To create a multiword BID, enclose the user batch ID in single quotation marks (' '). Single-word BIDs (with no spaces) do not require quotes. Wildcard specifications (like an asterisk, * or question mark, ?) are supported. If you use asterisk or wildcard on Solaris, you must escape these characters with a backslash such as: BID=*.
	To specify a batch number, use a pound sign followed by the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign. If you use a pound sign, on Solaris, it must be escaped by preceding it with a backslash such as:
	BID=\#14.
	If neither the **BID** nor the **ID** parameter is specified, all batches in the current mailbox are selected and concatenated into one file.
	You must specify both BID and ID. If you do not know the ID or BID, use an asterisk.
	If the Connect:Enterprise UNIX server you are connecting to is running on a Solaris operating system, you must escape the asterisk (*), question mark (?), and pound sign (#) characters by preceding them with a backslash. Following are two examples:
	`scp -P port# 'user@host:\$\$id=t\*\ bid=\#1' /tmp/flags4`
	`scp -P port# 'host@port:\$\$id=\new\?\ bid=\#1' /tmp/flags5`

ID=xxxxxxxx	identifies the 1–8 character mailbox ID of the remote site requesting the data. Wildcard specifications (like an asterisk, *) are supported. You must specify both BID and ID. If you do not know the ID or BID, use an asterisk.

Parameter	Description
FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I specifies to get all batches added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I requests the batches that are neither added locally nor incomplete. The syntax CM!I requests batches that were collected remotely, are multitransmittable, and are not incomplete.
FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for retrieval. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. Specifying FTIME=010101:1315 limits the command to batches created on or after January 1, 2002 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm Specifies on or after the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ONEBATCH=YES|Y|NO|N	indicates that only the first batch matching the selection criteria is to be transmitted to the remote site in response to the command. **YES|Y** – Specifies to transmit only a single batch to the remote site. If multiple files match the request, this value indicates only to send the first match. **NO|N** – Specifies to transmit all batches that conform to the request to the remote site. This is the default.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.

Parameter	Description
TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on or before which a batch must have been created to be eligible for retrieval. If the **FTIME** parameter is omitted, the creation date and time of oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **get** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **get** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20000101 lists all batches created on or between December 31, 1999 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. The entry TTIME=011231:1400 limits the **get** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm Specifies on or before the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

Extract Examples

The following example uses $$ keywords to extract batch01 from the user02 mailbox through port 9999 on cehost to the current directory on the local computer and changes the name to file01:

```
$ scp -P 9999 'user01@cehost:\$\$id=user02\ bid=batch01' file01
```

The following example uses $$ keywords to extract all batches beginning with *batch* from all mailboxes beginning with *user* through port 9999 on cehost to the current directory on the local computer and creates a corresponding file for each in the *download* directory.

```
$ scp -P 9999 'user01@cehost:\$\$id=user*\ bid=batch*' download
```

Using an SCP Windows Client

If you are using an SCP client on Windows, the behavior is determined by your client. The command parameters relevant to Connect:Enterprise are the same as those discussed in *Adding a File with Standard Syntax* on page 109. Because the Windows SCP clients are graphic interfaces, the command parameters are generally not visible to the user.

Chapter 7

Exchanging Files Using WebDAV Protocol

The Web-based Distributed Authoring and Versioning (WebDAV) protocol allows you to interact with Connect:Enterprise mailboxes using any WebDAV client.

This chapter describes the functionality of two WebDAV connection methods, including limitations and special considerations for each. Topics include:

- *Connecting to a Connect:Enterprise WebDAV Server* on page 119
- *Using Windows as a WebDAV Client* on page 121
- *Using a Command Line WebDAV Client* on page 123

Connecting to a Connect:Enterprise WebDAV Server

Every WebDAV client requires the following information to log on to a Connect:Enterprise WebDAV server:

- URL for the Connect:Enterprise WebDAV server
- Connect:Enterprise mailbox ID
- Connect:Enterprise password
- To use Secure FTP, both the sending and receiving sites must have FTP software that is capable of using Secure Sockets Layer (SSL) protocol. Refer to your client documentation for instructions on identifying your SSL certificate.

Connecting in Windows by Creating a Network Place

Connect to a Connect:Enterprise WebDAV server with Windows by creating a network place using the following procedure:

Note: You cannot create a network place until a batch is added to the mailbox that you want to open.

1. Open **My Network Places**.
2. Open **Add a Network Place**.

3. Type the URL to the Connect:Enterprise WebDAV server. You are prompted for a user ID and password.
4. Type your mailbox ID and associated password and click **OK**. The contents of your default mailbox is displayed. You default mailbox is defined on the Connect:Enterprise server.

After you connect, you can issue commands available to Windows. Refer to *Using Windows as a WebDAV Client* on page 121.

Connecting in Windows by Opening as a Web Folder

You can connect to a Connect:Enterprise WebDAV server using a Web browser. Use the following procedure for Microsoft Internet Explorer:

Note: You cannot open as a Web folder until a batch is added to the mailbox that you want to open.

1. Select **File**>**Open**.
2. Type the URL to the Connect:Enterprise WebDAV server.
3. Enable **Open as Web Folder** and click **OK**.
4. Type your mailbox ID and associated password and click **OK**.The contents of your default mailbox is displayed. You default mailbox is defined on the Connect:Enterprise server.

After you connect, you can access any Connect:Enterprise mailbox that you have access privileges for. You can navigate to other mailboxes using standard Windows features.

You can access the Connect:Enterprise WebDAV server as a network place using any Windows applications that will communicate with Microsoft BackOffice.

After you connect, you can issue commands available to Windows. Refer to *Using Windows as a WebDAV Client* on page 121.

Connecting Using a Command Line WebDAV Client

You can connect to a Connect:Enterprise WebDAV server using a command line WebDAV client. This is a more common method if you are using the UNIX operating system. The procedure you use to connect depends on the client that you are using. Use the following procedure for Cadaver:

1. Run Cadaver. The command prompt is displayed, similar to the following:

```
dav:!>
```

2. Type the URL to the Connect:Enterprise WebDAV server and press **Enter**. Following is an example:

```
dav:!> open http://host:port/ceudav
```

3. At the Username prompt, type your Connect:Enterprise mailbox ID and press **Enter**.

4. At the Password prompt, type your password. A command prompt is displayed that reflects the URL you are connected to. Following is an example:

```
dav:/ceudav/>
```

After you connect, you can issue commands available to your client. Refer to *Using a Command Line WebDAV Client* on page 123.

Using Windows as a WebDAV Client

You can use Windows as a WebDAV client to view the contents of a mailbox, search for a batch, and add and extract batches.

Viewing Contents of a Mailbox

To view the contents of a mailbox, connect to a Connect:Enterprise WebDAV server using a Web browser. Use the following procedure for Microsoft Internet Explorer:

1. Type the URL to the Connect:Enterprise WebDAV server in the **Address** field and click **OK**.

   A list of mailboxes that you have access to is displayed with the following information:

Column	Description
Mbx ID	Link to mailboxes that you have access to.
Created	Date and time stamp of the latest batch in the mailbox.
Orig ID	Mailbox ID of the person who originated the latest file in the mailbox.

2. Click a **Mbx ID**. A Mailbox Listing is displayed for the mailbox. This listing contains all batches in the mailbox and the following information on each batch:

Column	Description
Mbx ID	Mailbox ID of the current mailbox.
Batch No.	Batch number of the file. This number is assigned by Connect:Enterprise.
Batch ID	Batch ID of the file. This is similar to file name. There can be multiple batches with the same batch ID. They will have different batch numbers.
Size	Size of the batch in bytes.
Created	Date that the batch was added to the mailbox.
Status	Status flags associated with the batch.
Orig ID	Mailbox ID of the person who originated the batch.

The bottom of the page includes a count of the total number of batches and the options used to display them.

3. To return to the list of mailboxes, click **Back** on your browser.

   You cannot add or extract batches using the mailbox listing. To add or extract batches, refer to *Adding and Extracting Batches* on page 123.

Searching for a Batch

If the information displayed in the Mailbox Listing does not give you the information that you need, you can search a Connect:Enterprise mailbox for batches that match search criteria. Use the following procedure for Microsoft Internet Explorer:

1. Select **File**>**Open**.
2. Construct the URL in the **Address** field as follows:

```
http://host:port/ceudav/mailbox/?keyword1=value1&keyword2=value2
```

The following table describes the URL elements.

Element	Description
http://*host:port*/ceudav/	URL to the Connect:Enterprise
mailbox	The mailbox to search in. This is optional. If it is not included, Connect:Enterprise searches all mailboxes you have access to.
?	Indicates that search keyword are used in the command.
keyword1	The keyword for the first search criteria. Refer to *Available Keywords* on page 123.
value1	Value associated with keyword 1.
&	Indicates another search keyword. You can string together as many keywords as you need as long as they are preceded by the ampersand (&).
keyword2	The keyword for the next search criteria. Refer to *Available Keywords* on page 123.
value2	Value associated with keyword 2.

3. Click **OK**. The search returns a list of batches that match the search. You can extract the batches using any Windows method for moving or opening a file.

   Following is an example of a search entry:

```
http://host:port/ceudav/mailbox01/?&BID=batch01&BNO=122&MULTXMIT=N&OPT1
```

Available Keywords

The following table details the available search keywords. Keywords are not case-sensitive.

Keyword	Description
ID=*mailboxID*	Matches batches with the specified 1–8 character mailbox ID.
BID=*batchID*	Matches batches with the specified 1–64 byte user batch ID.
BNO=*batch#*	Matches batches with the specified batch number. This number is assigned to the batch by Connect:Enterprise so is not available for adding a batch.
CODE=A|E|B	Matches batches with the specified format: A = ASCII, E = EBCDIC, and B = BINARY. The default is ASCII.
ORIG=*mailboxID*	Searches on the originator of the batch. The originator is the user who originally put the batch in the repository.
FLAGS=	Searches by specific process flags. The definitions of all flags are listed in *Flags* on page 11. You can negate a flag by adding an exclamation point (!) in front of it. For example, A!I requests a directory listing for all batches that were added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I requests a directory listing of the batches that are neither added locally nor incomplete. The syntax CM!I specifies to list batches collected remotely, that are multitransmittable, and are not incomplete.

Adding and Extracting Batches

To add a batch from a mailbox, connect to the mailbox using *Connecting in Windows by Creating a Network Place* on page 119 or *Connecting in Windows by Opening as a Web Folder* on page 120. After connecting, you can drag and drop files between the mailbox and your computer.

When you add batches using Windows, you cannot specify parameters specific to Connect:Enterprise. This is a limitation of Windows. Because of this limitation, Connect:Enterprise assigns default values according to the FTP PUT options defined in your account definition. Contact the Connect:Enterprise administrator for more information.

Using a Command Line WebDAV Client

After you connect with a command line client, you can issue the commands available with your client. Connect:Enterprise may not support all commands available to your client. With Cadaver, for example, Connect:Enterprise supports only the following commands: ls, cd, pwd, put, get, mget, mput, propnames, propget, quit, lcd, lls, lpwd, logout, help, about.

You can use Windows as a WebDAV client to view the contents of a mailbox, search for a batch, and add and extract batches.

Viewing the Contents of a Mailbox

To view the contents of a mailbox:

1. Navigate to the mailbox you want to view using the **cd** command. You must be in a mailbox to issue commands. You cannot issue command from the ceudav> prompt.
2. Type **ls** to view the contents of the mailbox.

Viewing the Properties of a Batch

To view the properties of a batch:

1. Navigate to the mailbox you want to view using the **cd** command.
2. Type **ls** to view the contents of the mailbox.
3. This step is optional. Use it to display a list of available properties to view and a URL where these properties are described. Type the following command and press **Enter**:

```
propnames batchID
```

where *batchID* is batch ID you want to view the properties for. A list of available properties is displayed. Following is an example:

```
dav:/host/qatest/> propnames Batch01.txt
Fetching property names `Batch01.txt': DAV: creationdate
DAV: displayname
DAV: resourcetype
DAV: getcontentlength
DAV: getcontenttype
DAV: getlastmodified
www.sterlingcommerce.com/ceumailboxschema/batchnumber
www.sterlingcommerce.com/ceumailboxschema/ mailboxid
www.sterlingcommerce.com/ceumailboxschema/ dataformatflags
www.sterlingcommerce.com/ceumailboxschema/ processflags
www.sterlingcommerce.com/ceumailboxschema/ protocolflags
www.sterlingcommerce.com/ceumailboxschema/ originatorid
succeeded.
```

4. Type the following command and press **Enter**:

```
propget batchID
```

where *batchID* is batch ID you want to view the properties for. Following is an example:

```
dav:/host/qatest/> propget Batch01.txt
Fetching properties for `Batch01.txt':
creationdate = 2005/03/25-14:10
getlastmodified = Fri, 25 Mar 2005 20:10:47 GMT
displayname = Batch01.txt
resourcetype =
getcontentlength = 13
getcontenttype = application/octet-stream
batchnumber = 2
mailboxid = mailbox01
dataformatflags = IMAGE
processflags = COLLECTEDONLINE,REQUESTABLE,XMITYES
protocolflags = HTTP
originatorid = mailbox01
```

If there are multiple batches with the same batch ID, Connect:Enterprise returns a set of information for each batch.

Extracting Batches

The commands used to extract batches from Connect:Enterprise may be different for every client. For Cadaver, use **get** and **mget**. With these commands, you can specify search parameters. Connect:Enterprise will download batches that meet the search criteria.

If there are multiple batches with the same batch ID, and you use **get**, you will get all matching batches concatenated into a single batch. If you use **mget**, you get all matching batches individually. Use the following procedure:

1. Navigate to the mailbox you want to extract files from using the **cd** command. This step is optional. If you issue the command at the repository level (not in a mailbox), Connect:Enterprise searches all mailboxes that you have access to.
2. Construct the **get** or **mget** command as follows:

```
get batchID?keyword1=value1&keyword2=value2 localpath
```

The following table describes the elements of the command:

Element	Description
batchID	The batch ID to extract. This is required for an extract.
mailbox	The mailbox to search in. This is optional. If it is not included, Connect:Enterprise searches all mailboxes you have access to.
?	Indicates that search keyword are used in the command.
keyword1	The keyword for the first search criteria. Refer to *Available Keywords* on page 126.
value1	Value associated with keyword 1.

Element	Description
&	Indicates another search keyword. You can string together as many keywords as you need as long as they are preceded by the ampersand (&).
keyword2	The keyword for the next search criteria. Refer to *Available Keywords* on page 126.
value2	Value associated with keyword 2.
localpath	Path and file name where you want the batch to be saved. If you do not include this element, Connect:Enterprise uses the original batch ID and includes the entire **get** search string in the file name.

Note: If you include spaces in the localpath, you must enclose it in single quotes, and enclose the entire search string in double quotes. For example:

```
get "mailboxID/batchID?EO=Y&TO=Y"'new file'"
```

Available Keywords

The following table details the available search keywords. Key words are not case-sensitive.

Keyword	Description
BID=*batchID*	Matches batches with the specified 1–64 byte user batch ID
BNO=*batch#*	Matches batches with the specified batch number. This number is assigned to the batch by Connect:Enterprise. If you specify both BID and BNO, Connect:Enterprise filters on BNO, even if the value that you specify for BID does not match.
FLAGS=	Matches batches with specific process flags. The definitions of all flags are listed in *Flags* on page 11. You can negate a flag by adding an exclamation point (!) in front of it. For example, A!I requests a directory listing for all batches that were added locally with **cmuadd** (A flag), but are not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I requests a directory listing of the batches that are neither added locally nor incomplete. The syntax CM!I specifies to list batches collected remotely, that are multitransmittable, and are not incomplete.
ID=*mailboxID*	Matches batches with the specified 1–8 character mailbox ID.
ONEBATCH=Y|N	indicates that only the first batch matching the selection criteria is extracted. **Y** – Specifies to transmit only a single batch to the remote site. **N** – Specifies to transmit all batches that conform to the request to the remote site as a single file. This is the default.
ORIG=*mailboxID*	Matches batches based on the originator of the batch. The originator is the user who originally put the batch in the repository.

Adding Batches

The commands used to add batches from Connect:Enterprise may be different for every client. For Cadaver, use **put**. With this commands, you can specify parameters specific to Connect:Enterprise.

Use the following procedure:

1. Navigate to the mailbox you want to add files to using the **cd** command.
2. Construct the **put** command as follows:

```
put localpath mailboxID/batchID?keyword1=value1&keyword2=value2
```

The following table describes the elements of the command:

Element	Description
localpath	Path and file name of the file you want to add to the mailbox. If you don't specify a path, the current working directory on your local system is assumed.
mailboxID	Mailbox where you want the batch added. If you don't include the mailboxID, the current working mailbox is assumed.
batchID	Name that you want to give the batch. If you do not include this element, Connect:Enterprise uses the original file name and includes the entire **get** search string in the file name.
?	Indicates that additional parameters are used in the command.
keyword1	The keyword for the first parameter. Refer to *Available Keywords* on page 127.
value1	Value associated with keyword 1.
&	Indicates another keyword is used. You can string together as many keywords as you need as long as they are preceded by the ampersand (&).
keyword2	The keyword for the next parameter. Refer to *Available Keywords* on page 127.
value2	Value associated with keyword 2.

Note: If the file you are adding has spaces, you must enclose the file name in single quotes, and enclose the entire argument string in double quotes. For example:

```
put "'new file' mailboxID/batchID?EO=Y&TO=Y"
```

Available Keywords

The following table details the available search keywords. Keywords are not case-sensitive.

Keyword	Description
ID=*mailboxID*	Add the specified 1–8 character mailbox ID.

Keyword	Description
BID=*batchID*	Adds the specified 1–64 byte user batch ID
XMIT=Y|N	Indicates whether the batch is limited to host site use or can be distributed to other locations. **Y** – Batch cannot be transmitted. **N** – Batch can be transmitted. This is the default.
MULTXMIT=Y|N	Indicates whether the batch can be transmitted multiple times. **Y** – The batch can be transmitted multiple times. **N** – The batch cannot be transmitted multiple times. This is the default.
EO=Y|N	Indicates if a batch can only be extracted once. **Y** – The batch can only be extracted once. **N** – The batch can be extracted more than once. This is the default.
TO=Y|N	Indicates if a batch can only be transmitted once. **Y** – The batch can only be transmitted once. **N** – Search for batches that do not meet this description. This is the default.
CODE=A|E|B	Specifies the format of the batch. A = ASCII, E = EBCDIC, and B = BINARY. The default is ASCII.
TRIGGER=Y|N	Indicate to allow the file to be rerouted automatically. **Y** – Batches can be rerouted. **N** – Batches cannot be rerouted. This is the default.
ORIG=*mailboxID*	Specifies the originator of the batch. The originator is the user who originally put the batch in the repository.

Chapter 8

Async Protocol Commands

This chapter describes the following commands available to remote sites using Connect:Enterprise Async communications protocol:

- **$$ADD**—Sends a batch of data to the local site.
- **$$DELETE**—Flags a batch of data as deleted at the local site.
- **$$DIRECTORY**—Requests a formatted listing of batches from the local site.
- **$$REQUEST**—Requests a batch of data from the local site.

Before you issue any of these commands, verify that you have met the prerequisites listed in the following section.

Requirements

Obtain the following information from your local site system administrator prior to executing a Connect:Enterprise command:

- Mailbox ID
- Password and any additional site-specific security
- Access method (interactive session or Gateway emulation)
- Phone number and communications parameters (baud rate, number of data bits/stop bits, parity)

Four types of Async communications protocols are supported:

- XMODEM (checksum, cyclic redundancy check [CRC], and 1k)
- YMODEM Batch
- ZMODEM
- Kermit

Async communications software packages like ProComm enable users to configure the file overwrite option. This option controls the behavior of protocols like ZMODEM, when the remote side (in this case Connect:Enterprise) is trying to send a file with a name that already exists on the local PC (ProComm user). If the option is not set, ZMODEM informs Connect:Enterprise to skip

the file and the file is not transferred. Remote users should set this option so that the file is always transferred.

Remote users communicating with Connect:Enterprise at high speeds must turn on the hardware flow control option in their communications package. Remote sites using ZMODEM protocol must turn off the ZMODEM crash recovery option in their communications package.

Async Access Methods

For each Async port assigned to Connect:Enterprise, an access method can be defined by the local site administrator using the **Interactive port** parameter in the Async communications port definition file. This parameter is one of several that can be defined for each Async port at the local site. Unless the local site uses a hunt group to distribute incoming calls for a single phone number to multiple ports, a one-to-one relationship exists between a dialed phone number and a port defined in the local site Async communications port definition. For the dialed phone number, the remote site user must know the access method defined for that port.

Interactive Session Access Method

When the remote site uses the Interactive Session access method, the mailbox session is a two-way conversation established by dialing and logging in. After login, a Welcome message is transmitted to the remote site, and a command prompt is displayed (cmd>).

You can issue any valid $$ command at this prompt to control the session interactively, after you issue a **setx**, **sety**, **setz**, or **setk** command to specify the desired transmission protocol (XMODEM, YMODEM, ZMODEM, or Kermit). The Interactive Session access method is used for the command examples in this chapter.

Noninteractive Access Method

The noninteractive access method implements the $$ controls as follows:

- When the dial is complete, the remote site initiates an upload immediately. The first record of the uploaded file is a signon card in the following format:

```
/*SIGNON RMTID=xxxxxxxx PASSWORD=xxxxxxxx <CRLF>
```

- This record is followed immediately by one or more valid $$ commands that can be stacked before the data records that follow a **$$ADD** command. After connecting, the remote site uploads the following file:

```
/*SIGNON RMTID=ACME PASSWORD=letmein
$$REQUEST ID=ACME BATCHID='purchase orders'
$$DIR ID=ACME
$$ADD ID=ACME BATCHID='invoices'
...  invoice data record ...
...  invoice data record ...
...  invoice data record ...
...  invoice data record ...
```

- The remote site uploads the single file containing the signon card, $$ commands, and data to be added to the repository using the **$$ADD** command. Then the remote site initiates a download to receive the directory information and the requested purchase order batches. These actions occur during the same call.
- If the remote site does not supply a /*SIGNON card, then the first record must be a valid Connect:Enterprise $$ card. The $$ card is parsed with the ID used as the mailbox ID. If no password is set in the remote account definition file using the **Password** parameter, no further verification is done. If **Password** is set in the account definition, three tests are conducted to determine the password:

  a. Is a PASSWORD= parameter supplied on the $$ card? If so, the associated value is accepted as the password.

  b. Does the user batch ID contain a space? If so, all the characters up to the space or eight characters, whichever occurs first, are assumed to be the password. If more than eight characters occur before the space, then the password is truncated to eight characters. The BATCHID recorded for the batch is only those characters to the right of the first space. For example:

```
$$ADD ID=JOE BID='password THIS IS THE LITERAL'
```

  c. If no embedded space is present in the user batch ID, then the first eight characters of the user batch ID are considered the password. The BATCHID recorded for the batch is the entire original user batch ID, including the first eight characters. For example, when a password has been found, it is tested against the password in the account definition. If the password does not match, a disconnect occurs.

Note: When the Async daemon is running in noninteractive mode (the **Interactive port** checkbox in the communications port definition is not checked), it is not necessary for a remote site to have a $$ card in their data. The data received without a **$$ADD** card is added to the repository with mailboxID=devicename, where *devicename* is the name of the device where the data is received. The Connect:Enterprise site must have an account definition file named *devicename* with the **Password** parameter set to *devicename.*

Noninteractive access method supports the following four Async protocols: ZMODEM, XMODEM, YMODEM, and Kermit. When ZMODEM or Kermit is used in Noninteractive access method, the response to **$$DIR** and **$$DEL** cards is sent to the Async remote site in files named DIR.LOG and DEL.LOG, respectively. When XMODEM protocol is used in Noninteractive access method, the remote user selects the file names that receive the output of **$$DIR** or **$$DEL**.

Note: In order for the **$$DELETE** and **$$DIRECTORY** commands to be used in Noninteractive access method, the **Remote communication sequence (mode)** parameter must be set to **Send, then receive**.

Noninteractive Remote Sites Prefixing Data with $$ Cards

When an Async device is configured in the async communications port definition as a noninteractive device, and remote sites need to use flags set with $$ cards, they must prefix their data with the $$ card. Async protocol parses the first 256 bytes of the incoming data for $$ cards.

All $$ cards must be terminated either with a single carriage return line feed (CRLF) pair or a single line feed (LF). All extra CRLFs or LFs are treated as part of the data. Remote sites sending binary data must create a separate text file containing a $$ card and concatenate the text file and the binary file into one file.

On UNIX systems, use the following command:

```
cat  card_file binary_data_file  >  some_file
```

On DOS systems, use the following command:

```
copy  /b  card_file +  binary_data_file  some_file
```

When you create card_file (using either vi on UNIX or edit on MS-DOS) do not press **Enter** at the end of your card because a CRLF is automatically added to terminate the $$ card.

Logging On

To log on to Connect:Enterprise at a port using the Interactive Session access method, use this procedure:

1. Dial the local site Connect:Enterprise number. A login prompt is displayed.
2. Enter the mailbox ID for the location from which you are sending. You are prompted for a password.
3. Supply a password and press **Enter**. The following prompt is displayed:

```
cmd>
```

Refer to the *Noninteractive Access Method* on page 130 for more information on noninteractive login.

Selecting an Async Transmission Protocol During an Interactive Session

The default protocol for a port is defined by the system administrator using the **Protocol** parameter in the communications port definition file. Refer to the *Connect:Enterprise UNIX Installation and Administration Guide* for further details. To specify a different protocol, use the **set?** command, where *?* is one of four options:

Option	Description
setx	XMODEM
sety	YMODEM
setz	ZMODEM
setk	Kermit

To change the protocol for this session, use the following format:

```
cmd> setx|sety|setz|setk
```

The command prompt is provided only for ports with the communications port definition parameter **Interactive port** enabled.

Press **Enter** to confirm your protocol choice. The UNIX Async Server uses the protocol you specified for your file transfer requests. At this point, you can enter any valid mailbox command.

Caution: If you press **ESC** during an Async interactive session, your computer might stop responding. Pressing **CTRL+X** ends the session and resolve the problem.

Logging Off

When you finish your interactive mode Connect:Enterprise session, log off using the following command:

```
$$logoff
```

In noninteractive mode, logoff is automatic when the remote user terminates the communication session. To force Connect:Enterprise to disconnect the session after the **$$ADD** and **$$REQ** commands have executed, use the **$$LOGOFF** command in the upload file as in the following example:

```
/*SIGNON RMTID=ACME PASSWORD=letmein
$$REQ BID='reports1' CONV=A RECVFILE='report1.txt'
$$REQ BID='reports2' RECVFILE='report2.bin'
$$LOGOFF
$$ADD ID=ACME BATCHID='invoices'
...  invoice data record ...
...  invoice data record ...
...  invoice data record ...
...  invoice data record ...
```

$$ADD Command

The **$$ADD** command is used to add data to the repository. When a **$$ADD** command is issued in Async interactive mode, Connect:Enterprise returns a single record upon successful completion. This record is in the same format as those returned when a **$$DIR** command is issued. The record confirms the creation of a new batch, providing information such as the batch number and the status flags. For the format of this record, refer to the example shown for the **$$DIR** command in the *$$DIRECTORY Command* on page 143.

The permissions for your mailbox ID are set up by the system administrator in the *mbxacl.conf* file in the *$CMUHOME/med* directory. This file controls which mailboxes you can add batches to.

$$ADD Command Format

All parameters are optional.

You can use any number of blanks between parameters. However, blanks are not allowed within parameters, with the exception of the BATCHID parameter. In noninteractive Async mode, the **$$ADD** keyword must begin in column 1.

Command	Parameter
$$ADD	BATCHID='xx...xx'
	CODE=A|E|B
	EO=YES|Y|NO|N
	ID=XXXXXXXX
	MULTXMIT=YES|Y|NO|N
	PASSWORD=XXXXXXXX
	TO=YES|Y|NO|N
	TRIGGER=YES|Y|NO|N
	XMIT=YES|Y|NO|N

The following table describes the available parameters.

Parameter	Description
BATCHID='xx...xx' (or BID)	identifies the 1–64 character user batch ID for the batch being added. It must be enclosed in either single or double quotes if it contains spaces. In noninteractive mode, when a remote user sends a **$$REQ** card without previously sending a SIGNON card and PASSWORD on the **$$REQ** command, the Async server extracts the password from the first word of the batch ID or the first eight characters, if the first word is more than eight characters. For example, if the BID is specified as 'ASCII test file', the word ASCII is the password and the remaining string, 'test file', is the BID. The extract occurs with BID set to 'test file'.
CODE=A|E|B	The CODE parameter identifies the data format of the batch being added. **A**—The batch contains ASCII data. This is the default. **E**—The batch contains EBCDIC data. **B**—The batch contains binary data.
EO=YES|Y|NO|N	**YES|Y**—specifies that the batch is only allowed to be locally extracted once and is flagged nontransmittable. After local extraction, the batch is permanently locked and flagged as nontransmittable and unextractable.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your remote site. If omitted, the ID defaults to the resource name. If BATCH security is being enforced, the ID must be one of the mailbox IDs contained in the **Valid ID list** parameter in the MED file.
MULTXMIT=YES|Y|NO|N	The **MULTXMIT** parameter indicates whether the batch is available for multiple transmissions. **YES|Y**—Indicates that the batch can be transmitted multiple times. If YES is specified, it overrides the **XMIT** parameter, setting it to YES. When MULTXMIT=Y, the added batch is flagged as multitransmittable when it is added to the mailbox. The batch bearing a MULTXMIT=Y parameter is not flagged as transmitted when successfully transmitted. **NO|N**—Indicates that the batch cannot be transmitted multiple times. This is the default.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.
TO=YES|Y|NO|N	**YES|Y**—specifies that the batch can only be remotely transmitted once and cannot be locally extracted; it is flagged as unextractable. After transmission to the intended remote site, the batch is permanently locked and then flagged as nontransmittable and unextractable. If transmission of a TO=Y batch fails when one or more records have been transmitted, the batch is still locked. To retry the transmission, a new batch must be added from the original source.

Parameter	Description
TRIGGER=YES|Y|NO|N	immediately reroutes files to other remote sites. In order for automatic routing to function, an Auto Connect file must be defined with the **Contact** parameter set to **Forward data to Remote site automatically**. When the characteristics of the batch being added match the selection criteria in this ACD file, the batch is automatically forwarded to the destination specified in the ACD file. Refer to the *System Configuration* chapter of the *Connect:Enterprise UNIX Installation and Administration Guide* for more information about the ACD file. **YES|Y**—The batch is rerouted if a valid Auto Connect list with matching selection criteria has been defined. **NO|N**—The batch is not forwarded. This is the default.
XMIT=YES|Y|NO|N	indicates whether the batch is available for transmission to another remote site, after it is successfully added to the mailbox. **YES|Y**—Indicates that the batch is transmittable to any remote site that identifies the correct mailbox ID. A requestable flag is set. When XMIT=Y, the added batch is flagged as transmitted after it is successfully sent to another remote site. **NO|N**—Indicates that the batch is not transmittable. The batch can only be extracted at the local site. This is the default.

Examples

The following examples illustrate a remote site (ACME) using the Interactive Session access method. ACME issues the **$$ADD** commands after establishing a session with the Connect:Enterprise Async Server by dialing, logging on, and issuing a **setx, sety, setz,** or **setk** command to specify an appropriate transmission protocol (XMODEM, YMODEM, ZMODEM, or Kermit). The **$$ADD** command syntax is the same for users of the Noninteractive access method, but the command is transmitted within the data file with other $$ commands, if any, and the required /*SIGNON card.

ACME transmits a report file to a Connect:Enterprise local site. The data is in ASCII format and is identified with the user batch ID of 'sales summary'. ACME intends the final destination of the data to be the local site and chooses the defaults XMIT=N, MULTXMIT=N, and TO=N. ACME does not want to limit the local site to a single extraction so it accepts the default EO=N. After issuing the **$$ADD** command at the command prompt, ACME initiates a protocol upload to transmit data to the local site mailbox. The batch is flagged as collected to indicate collection from a remote site. The batch does not have a requestable flag. It can be extracted by a local site **cmuextract** command, but without a requestable flag, it cannot be forwarded to another remote site.

```
cmd> $$ADD ID=ACME BID='sales summary'
```

Another ASCII file is destined for a single remote site named SMITH after being successfully transmitted to the mailbox. Use of the **XMIT=Y** parameter instructs Connect:Enterprise to allow the batch to be requested by another remote site. After issuing the **$$ADD** command at the command prompt, ACME initiates a protocol upload to transmit data to the local site mailbox. If

the transmission to the other remote site is unsuccessful, a transmitted flag is not posted on the batch, and it remains available for another attempt. A transmitted flag is posted on the batch after it is transmitted successfully and the requestable flag is retained. The transmitted flag makes the batch ineligible for additional transmissions, with one exception: after a transmitted flag is posted, it is still available to a remote site that uses a **$$REQUEST** command specifying the user batch ID.

Although ACME intends to make the batch available for forwarding to another remote site, they have chosen to permit local site extraction of the batch with **cmuextract** (TO=N is the default). Extraction at the local site posts an extracted flag, but this does not affect availability of the batch for transmission to another remote site unless the local site **cmuextract** operation flags the batch as deleted by using the DELETE=YES option.

This example also shows that ACME uses the ID of SMITH because the remote site SMITH specifies ID=SMITH on the **$$REQUEST** card to collect the available batch. ACME could have used ID=ACME, but then the SMITH remote site would have to specify **$$REQUEST** ID=ACME. Either way works, but the two remote sites should coordinate the use of mailbox IDs, taking into consideration the security administration at the local site.

```
cmd> $$ADD ID=SMITH BID='termination notices' XMIT=Y
```

The same company has a binary file it intends to forward to multiple remote sites through the Connect:Enterprise local site. After issuing the **$$ADD** command at the command prompt, ACME initiates a protocol upload to transmit data to the local site repository. The batch is flagged as collected, requestable, and transparent to indicate that Bisync stations consider this data to be binary. The batch is not flagged as transmitted upon retransmission from the local site mailbox because of MULTXMIT. The multitransmittable and requestable flags remain after transmission. The only action a remote site can take to make this batch unavailable for transmission is to logically delete it with a **$$DELETE** command. Without deleted or transmitted flags, a multitransmittable batch remains permanently eligible for transmission.

```
cmd> $$ADD ID=ACME BID='holiday greetings' MULTXMIT=Y CODE=B
```

To prevent local site extraction yet make an EBCDIC batch available for a single transmission to the remote site SMITH, TO=Y can be used to indicate a transmit once/transmit only status for the added batch. After issuing the **$$ADD** command at the command prompt, ACME initiates a protocol upload to transmit data to the local site repository. The batch displays the collected, requestable, and unextractable flags when the add is complete. Unextractable batches are permanently locked against local site extraction and following transmission to a remote site, the unextractable batch is flagged nontransmittable and transmitted.

Note: If transmission of this batch fails midway, it is flagged as nontransmittable, but not transmitted. This flag indicates a failed transmission, and this batch cannot be retransmitted. After a nontransmittable flag is set, the batch is permanently locked, and the originating station has to add a duplicate batch to the repository to retry the operation.

```
cmd> $$ADD ID=SMITH BID='for your eyes only' TO=Y CODE=E
```

ACME transmits a report file to the local site. The data is in EBCDIC format, and is identified with the user batch ID of 'sales summary'. ACME, the remote site, does not want the batch forwarded to other remote sites nor extracted at the local site more than once. The **CODE** parameter must be used (CODE=E) because the data format is not the default, ASCII. After issuing the **$$ADD** command at the command prompt, ACME initiates a protocol upload to transmit data to the local site repository. Connect:Enterprise flags the batch with a collected but not a requestable flag. The **EO=Y** parameter flags the batch with a nontransmittable flag to indicate extract only once. This batch is permanently locked against transmission, and upon successful extraction is locked with an unextractable flag.

```
cmd> $$ADD ID=ACME BID='sales summary' EO=Y CODE=E
```

The following example demonstrates transmission of a batch to multiple mailboxes in a single command. The following command generates a multi-add of the file.

```
cmd> $$ADD ID=A,B,C BATCHID= 'my_file'
```

Invalid $$ADD Parameter Combinations

The following combinations of parameters are invalid:

- Do not use XMIT=Y with MULTXMIT=Y.
- Do not use XMIT=Y with TO=Y.
- Do not use XMIT=Y with EO=Y.
- Do not use TO=Y with MULTXMIT=Y.
- Do not use EO=Y with MULTXMIT=Y.
- Do not use EO=Y with TO=Y.

Automatic Routing

Refer to the *System Configuration* chapter of the *Connect:Enterprise UNIX Installation and Administration Guide* for a description of Automatic Routing, which automatically forwards batches to remote sites as soon as they are added into a mailbox, for example, with a **$$ADD** command.

$$DELETE Command

The **$$DELETE** command marks a specific batch of data as logically deleted from the repository, using the D flag. The command can be abbreviated to **$$DEL**.

The permissions for your mailbox ID are set up by the system administrator in the *mbxacl.conf* file in the *$CMUHOME/med* directory. This file controls which mailboxes you can delete batches from.

After Connect:Enterprise receives the **$$DEL** command, it processes the command and returns the results to the remote site. The remote site collects the output to the console.

When a batch is logically deleted, the delete flag (D) is set. The local site operator can restore the batch (using the **cmustatus** command) or physically erase it (using the **cmuerase** command).

If the remote site is using the Interactive Session access method, Connect:Enterprise returns confirmation messages to the remote console after processing the **$$DELETE** command.

$$DELETE Command Format

Required parameters are in bold font in the following table. Optional parameters are in plain font. Parameters are listed alphabetically.

Any number of blanks can be used between parameters, but not within parameters. In noninteractive Async mode, the **$$DELETE** keyword must begin in the first record position. The **$$DELETE** command can be abbreviated to **$$DEL**.

Command	Parameter
$$DELETE	**BATCHID='xx...xx'|#nnnnnnnn**
	FLAGS=[[!][A|C|D|E|I|M|T|B|F|G|Q|Y|Z|K|R]]...
	FTIME=[[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ORIG=XXXXXXXX
	PASSWORD=XXXXXXXX
	RECEIPT
	RECVFILE=filename
	TTIME=[[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
BATCHID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. The user batch ID can be 1–64 characters. It must be enclosed in either single or double quotes if it contains spaces. Wildcard specifications (like an asterisk, *) are allowed. If a batch number is specified, a pound sign (#) must precede the (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign, separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.

Parameter	Description
FLAGS=[[!][A|C|D|E|I|M|T|B|F|G|Q|Y|Z|K|R]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of these flags. You can negate a flag by adding an exclamation point (!) in front of it. For example, A!I deletes all batches added locally with the **cmuadd** command (A flag), but not flagged incomplete (I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax!A!I deletes the batches neither added locally nor flagged incomplete. The syntax CM!I deletes batches collected remotely that are multitransmittable and are not flagged incomplete.
FTIME=[[CC]yymmdd[:hhmm]|nn[:hhmm]]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for deletion. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=010101 limits the **$$DELETE** command to those batches created on or after January 1, 2001. Specifying FTIME=25 also limits the **$$DELETE** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 deletes all batches created on or between December 31, 2001, and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. Specifying FTIME=010101:1315 limits the **$$DELETE** command to those batches created on or after January 1, 2001, at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow. The status report indicating which batches were logically deleted is written to standard output. If an error is encountered when the command line is parsed, the error messages are written to standard error.

Parameter	Description
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your remote site. The **ID** parameter is optional unless a user batch ID is entered. If omitted, the ID defaults to the resource name. If BATCH security is being enforced, the ID must be one of the mailbox IDs contained in the **Valid ID list** parameter in the MCD file. Wildcard specifications (like an asterisk, *) are allowed.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise remote command exit for those sites that have enabled custom security.
RECEIPT	causes Connect:Enterprise to reset the log maintained in the batch called <<ACTIVITY LOG>>. This parameter cannot be used with any other parameter. (Example: $$DELETE RECEIPT).

Parameter	Description
TTIME=[[CC]yymmdd[:hhmm] |nnn[:hhmm]]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on which or before which a batch must have been created to be eligible for deletion. If the **FTIME** parameter is omitted, the creation date and time of oldest batch are assumed as the start of the period. Specifying TTIME=011231 limits the **$$DELETE** command to batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **$$DELETE** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20020101 deletes all batches created on or between December 31, 2001, and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **$$DELETE** command to those batches created on or before December 31, 2001, at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. For example, TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow. The status report indicating which batches were logically deleted is written to standard output. If an error is encountered when the command line is parsed, the error messages are written to standard error.

Examples

The following examples illustrate ACME using the Interactive Session access method; therefore, ACME receives confirmation messages at its Async console when Connect:Enterprise processes the **$$DELETE** commands.

ACME deletes a specific batch by designating its batch number. ID is optional for processing the **$$DELETE** command. This following example illustrates using the command to logically delete batch number 775 and post a deleted flag. Deleted batches are not available for transmission, even if the requestable flag is on, and a batch number is specified in the **$$REQUEST** command. Deleted batches can be extracted at the local site, if the batch number is specified with the **cmuextract** command. Batches can be restored only at the local site using the **cmustatus** command. The batch is unrecoverable, within the scope of Connect:Enterprise, after a local site **cmuerase** command physically erases it.

```
cmd> $$DELETE ID=ACME BATCHID=#775
```

The following command turns on the deleted flag for every batch with the mailbox ID ACME.

```
cmd> $$DELETE ID=ACME
```

ACME narrows the deletion of ACME batches to those with user batch ID ’invoices’ and the mailbox ID ACME.

```
cmd> $$DELETE ID=ACME BID=’invoices’
```

ACME physically deletes the log batch maintained by Connect:Enterprise for ACME (called <<ACTIVITY LOG>>). This batch contains the log of ADD/REQ operations performed by ACME. This batch is maintained only if Connect:Enterprise has enabled the verification of the data delivery feature for the remote.

```
cmd> $$DELETE RECEIPT
```

After the log batch is deleted, the next add or extract of batches creates a new log batch for the remote user.

$$DIRECTORY Command

The **$$DIRECTORY** command displays the contents of the designated mailbox and batch status information. Batch selection is designated by mailbox ID alone, mailbox ID and user batch ID, by batch number, originator, start and end times, and flags.

The permissions for your mailbox ID are set up by the system administrator in the *mbxacl.conf* file in the *$CMUHOME/med* directory. This file controls which mailboxes you can obtain directory listings from. If you enter a **$$DIR** command for a mailbox to which you do not have access, you receive the message *Permission denied*.

The mailbox contents are displayed either to standard output or to a local file if one is specified as an argument.

$$DIRECTORY Command Format

Required parameters are in bold font in the following table. Optional parameters are in plain font. Both types of parameters are listed alphabetically.

You can use any number of blanks between parameters. However, blanks are not allowed within parameters, with the exception of the **BATCHID** parameter. In noninteractive Async mode, the **$$DIRECTORY** keyword must begin in the first record position. The **$$DIRECTORY** command can be abbreviated to **$$DIR**.

Command	Parameter
$$DIRECTORY	**BATCHID='xx...xx'|#nnnnnnnn**
	ID=XXXXXXXX
	FLAGS=[[!][A|C|D|E|I|M|T|B|F|G|Q|Y|Z|K|R]]...
	FTIME=[[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ORIG=XXXXXXXX
	PASSWORD=XXXXXXXX
	RECEIPT
	TTIME=[[CC]yymmdd[:hhmm]|nnn[:hhmm]]

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
BATCHID='xx..xx' | #nnnnnnnn	The BATCHID parameter is either the user batch ID or the batch number. The user batch ID can be 1–64 characters. It must be enclosed in either single or double quotes if it contains spaces. Wildcard specifications (like an asterisk, *) are allowed. If a batch number is specified, a pound sign must precede the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.

Parameter	Description
ID=xxxxxxxx	identifies the 1–8 character mailbox ID for which directory information is desired. Connect:Enterprise searches the repository for batches that match the specified mailbox ID. Wildcard specifications (like an asterisk, *) are allowed.
FLAGS=[[!][A|C|D|E|I|M|T|B|F|G|Q|Y|Z|K|R]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of these flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I obtains a directory listing for batches added locally with the **cmuadd** command (A flag), but not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I specifies a directory listing of the batches neither added locally nor flagged incomplete. The syntax CM!I request batches collected remotely that are multitransmittable, and are not flagged incomplete.

Parameter	Description
FTIME=[[CC]yymmdd[:hhmm]|nnn [:hhmm]]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for reporting. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME= 010101 limits the **$$DIR** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **$$DIR** command to those batches created on or after January 1, 2002, assuming today's date is January 26, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001, and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. Specifying FTIME=010101:1315 limits the **$$DIR** command to those batches created on or after January 1, 2001, at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. For example, FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for reporting. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.
RECEIPT	lists the directory record of the log batch maintained for the remote site. This log batch, called the <<ACTIVITY LOG>>, contains the log of ADD/REQ operations performed by the remote site. This parameter cannot be used with any other parameter. (Example: $$DIR RECEIPT).

Parameter	Description
TTIME=[[CC]yymmdd[:hhmm]|nnn [:hhmm]]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on which or before which a batch must have been created to be eligible for reporting. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the start of the period. Specifying TTIME=011231 limits the **$$DIRECTORY** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **$$DIRECTORY** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20020101 lists all batches created on or between December 31, 2001, and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **$$DIRECTORY** command to those batches created on or before December 31, 2001, at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. For example, TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for reporting. The possible valid combinations are as follows. ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

Output

After a **$$DIR** command is issued, Connect:Enterprise looks in the repository for the designated batches and returns that information to the remote site in the following format:

```
xxxxxxxx nnnnnnnn bbbbbbbbbb <x...x> YYMMDD-HHMM flags protocol format
```

The following table describes the output:

Parameter	Description
xxxxxxxx	The 8-character mailbox ID for the batch.
nnnnnnnn	The 8-digit batch number assigned to the batch.
bbbbbbbbbb	The total byte count of the batch.
<x...x>	The first 24 characters of the user batch ID. If the user batch ID is longer than 24 characters it is shortened to the first 23 characters of the literal with the twenty-fourth character becoming a plus sign (+).
YYMMDD–HHMM	The date and time that the batch is added to the repository.
flags	One or more flags identifying the current status of a batch in the repository. Refer to the **cmustatus** command in the *Connect:Enterprise UNIX Installation and Administration Guide, Administrator Commands* and *Operator Commands* chapters for more information.
protocol	indicates the transmission protocol used to place the batch into the specified mailbox. The following options are possible: ◆ TCP Transmitted locally with **cmuadd** ◆ ASY Transmitted remotely with Async protocol ◆ BSC Transmitted remotely with Bisync protocol ◆ FTP Transmitted remotely with FTP protocol ◆ FTS Transmitted remotely with Secure FTP protocol ◆ (blank) Activity log batch
format	indicates the data format of the batch. Valid values are ASC (ASCII), BIN (binary, or EBC (EBCDIC).

Examples

The examples in this section assume ACME is using the Interactive Session access method. The sample **$$DIRECTORY** commands are issued after a session with the Connect:Enterprise Async Server is established by dialing, logging in, and issuing a **setx, sety, setz,** or **setk** command to specify the appropriate transmission protocol (XMODEM, YMODEM, ZMODEM, or Kermit). The **$$DIRECTORY** command syntax is the same for users of the SPC Emulation access method, but the command is transmitted within the data file, as an XMODEM upload to the local site, along with other $$ commands, if any, and the required /*SIGNON card.

ACME requests a listing of batches stored in the mailbox with ID=ACME.

```
cmd> $$DIR ID=ACME PASSWORD=letmein
```

ACME narrows the selection of batches to those with the user batch ID ’orders’. When a user batch ID is specified, a mailbox ID is required.

```
cmd> $$DIR ID=ACME BID='orders' PASSWORD=letmein
```

ACME investigates the status flags for a specific, previously obtained batch number.

```
cmd> $$DIR ID=ACME BID=#787 PASSWORD=letmein
```

ACME requests the directory information for the log batch maintained by Connect:Enterprise for ACME (called <<ACTIVITY LOG>>). This batch contains the log of ADD/REQ operations performed by ACME. This batch is maintained only if Connect:Enterprise has enabled the verification of the data delivery feature for the remote site.

```
cmd> $$DIR RECEIPT
```

In the following example, the command returns all batches whose originator ID is the *originator name* indicated (within the constraints set forth by the ACL defaults section). If the current directory has no batches available to the current user (based on user permissions), then the **DIR** command does not return any batches.

```
cmd> $$DIR "$$ ORIG=originator name"
```

The following command gives a directory listing of all batches originated by Dave on or after July 22, 2002, 12:30.

```
cmd> $$DIR "$$ORIG=dave FROM=20020722:1230"
```

The **FLAGS** parameter is a filtering identifier.

```
cmd> $$DIR "$$ [FLAGS=char1[char2...]]"
```

The following example returns all batches flagged for multiple transmission (M) and as requestable (R).

```
cmd> $$DIR "$$ FLAGS=MR"
```

$$REQUEST Command

Data transmission from the local site is initiated by sending a **$$REQUEST** command. The **$$REQUEST** command can specify a single batch number, all batches for the specified mailbox

ID, or batches that have a specific mailbox ID and match a specified user batch ID (alphanumeric string). The **BATCHID** parameter can contain wildcard specifications to further define batches the remote site selects.

When multiple batches are requested (or the potential for multiple batch transmission exists), only batches not already marked as transmitted are selected. However, when a specific batch is requested by batch number, it is transmitted regardless of its transmission status. Further restrictions can be placed on the batches transmitted to provide automatic character set translation and record-reformatting capabilities.

The permissions for your mailbox ID are set up by the system administrator in the *mbxacl.conf* file in the *$CMUHOME/med* directory. This file controls the mailboxes from which you can retrieve batches.

$$REQUEST Command Format

Required parameters are in bold font in the following table. Optional parameters are in plain font.

You can use any number of blanks between parameters. However, blanks are not allowed within parameters, except for the **BATCHID** parameter. In noninteractive Async mode, the **$$REQUEST** keyword must begin in the first position. The **$$REQUEST** command can be abbreviated to **$$REQ**.

Note: If you are transmitting **$$ADD** and **$$REQUEST** cards at the same time, send the **$$REQUEST** cards first.

Command	Parameter
$$REQUEST	**BATCHID='xx...xx'|#nnnnnnnn**
	ID=XXXXXXXX
	BCHSEP=NO|OPT3|OPT4
	CONV=A|E|N
	FLAGS=[[!][A|C|D|E|I|M|T|B|F|G|Q|Y|Z|K|R]]...
	FTIME=[[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ONEBATCH=YES|Y|NO|N
	ORIG=XXXXXXXX
	PASSWORD=
	RECEIPTXXXXXXXX
	RECSEP=CR|CRLF|LF
	RECVFILE='xxx'
	TTIME=[[CC]yymmdd[:hhmm]|nnn[:hhmm]]

Required Parameters

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
BATCHID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. The user batch ID can be 1–64 characters. It must be enclosed in either single or double quotes if it contains spaces. Wildcard specifications (like an asterisk, *) are allowed. If a batch number is specified, a pound sign must precede the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your site. If batch level security is created in the Connect:Enterprise MCD file, the ID must be one of the IDs contained in the MCD **Valid ID list** parameter file. Wildcard characters (like an asterisk, *) are allowed.
BCHSEP=NO|OPT3|OPT4	specifies the batch separation method used to send data to the remote site. Options are: **NO|N**—Specifies that batches are not separated. If multiple batches are sent in a single connection, they are concatenated and sent as a single batch. Batches are flagged as transmitted throughout the transmission as each batch is successfully transmitted. **OPT3**—Specifies that batches are not separated. If multiple batches are sent in a single connection, they are concatenated and sent as a single batch. Batches are not marked as transmitted until all the batches matching the **$$REQUEST** command criteria are transmitted successfully. **OPT4**—Batches are separated if the protocol supports multiple file transfers (ZMODEM, YMODEM Batch, Kermit). The batches are named with the format *batch#.dat*. If the protocol does not support multiple file transfers, a separate invocation of the file transfer is made for each batch requested, and the remote site must be prepared to receive each file separately. Connect:Enterprise obtains default values from the account definition.

Parameter	Description
CONV=A|E|N	specifies the character code set in which to transmit data to the remote site. Because some conversions are not allowed (binary to ASCII for example), this parameter can limit the batches selected for transmission. The **CONV** parameter determines what type of record conversion Connect:Enterprise performs, if any. CONV processing rules can be specified in the account definition and overridden in the remote block of a schedule using the **Outbound data format** parameter. The default is CONV=N. **A**—Specifies ASCII data. Batches marked as containing ASCII or EBCDIC data and meeting the **$$REQ** requirements are transmitted to the remote site. Selected EBCDIC batches are translated to ASCII based on user or Sterling Commerce-supplied translation tables (specified by the account or schedule). No binary batches can be transmitted. **E**—Specifies EBCDIC data. Batches marked as containing ASCII or EBCDIC data and meeting the **$$REQ** requirements are transmitted to the remote site. ASCII records are translated to EBCDIC through user or Sterling Commerce-supplied translation tables. No binary batches can be transmitted. **N**—Specifies no translation. All batches that meet the **$$REQ** command requirements are transmitted to the remote site without translation.
FLAGS=[[!][A|C|D|E|I|M|T|B|F|G|Q|Y|Z|K|R]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of these flags. You can negate a flag by adding an exclamation point (!) in front of it. For example, A!I requests the added (A) batches but none of the incomplete (I) batches. These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I requests the batches neither added locally nor flagged incomplete. The syntax CM!I requests batches collected remotely that are multitransmittable and not flagged incomplete.

Parameter	Description
FTIME=[[CC]yymmdd[:hhmm]|nnn[:hhmm]]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for retrieval. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **$$REQUEST** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **$$REQUEST** command to those batches created on or after January 1, 2002, assuming that today's date is January 26, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001, and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. Specifying FTIME=020101:1315 limits the **$$REQUEST** command to those batches created on or after January 1, 2002, at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. Specifying FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd—Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm—Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn—Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm—Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm—Specifies on or after the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ONEBATCH=YES|Y|NO|N	indicates that only the first batch matching the selection criteria is transmitted to the remote site in response to the **$$REQ** command. **YES|Y**—When BID= is specified, this parameter indicates that only the first matching batch is returned. **NO|N—**Specifies that all batches that conform to the request are transmitted to the remote site. This is the default.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise Remote Command Exit for those mailbox sites that have enabled custom security.

Parameter	Description
RECEIPT	causes Connect:Enterprise to send a log batch called the <<ACTIVITY LOG>>. It is a log of all the ADD/REQ operations performed by the remote site. This batch can be considered as a receipt by the remote site. This batch is reset by the command **$$DELETE** RECEIPT. This parameter cannot be used with any other parameter.
RECSEP=CR|CRLF|LF	enables a remote site to specify the desired record separator character(s) to be used by the local site when the requested batch is transmitted, overriding the RECSEP definition made at the local site, if necessary. **CR**—Specifies record separation by carriage returns (hex ’0D’). **CRLF**—Specifies record separation by carriage return/line feed pairs (hex ‘0D0A’). **LF**—Specifies record separation by line feeds (hex ’0A’).
RECVFILE=‘xxxxxxxxxxx’	a file name specified by the remote Async user on the command line (or on a **$$REQ** card) when a batch is downloaded. File names used with the **RECVFILE** parameter can have 1–12 characters and must be enclosed in single quotation marks. The following example illustrates the use of RECVFILE on an interactive **$$REQ** command: $$req id=mailbox bid=‘November Reports‘ recvfile=‘reports.nov‘ RECVFILE is valid for BCHSEP=No and BCHSEP=OPT3. If the remote user specifies both RECVFILE=‘*something*’ and BCHSEP=OPT4, Connect:Enterprise generates a CMUError 0x1010, because RECVFILE and BCHSEP are mutually exclusive. In noninteractive mode, RECVFILE can be specified for **$$DIR** and **$$DEL** commands.

Parameter	Description
TTIME=[[CC]yymmdd[:hhmm] |nnn[:hhmm]]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date on which or before which a batch must have been created to be eligible for retrieval. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the start of the period. Specifying TTIME=011231 limits the **$$REQUEST** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **$$REQUEST** command to those batches created on or before December 31, 2001, assuming that today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20020101 lists all batches created on or between December 31, 2001, and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **$$REQUEST** command to those batches created on or before December 31, 2001, at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. For example, TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd—Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm—Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn—Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm—Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm—Specifies on or before the time hhmm today **Note:** Do not use the **TTIME** parameter without the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

Examples

The examples in this section assume ACME is using the Interactive Session access method. The sample **$$REQUEST** commands are issued after a session with the Connect:Enterprise Async Server is established by dialing, logging in, and issuing a **setx, sety, setz,** or **setk** command to specify the appropriate transmission protocol (XMODEM, YMODEM, ZMODEM, or Kermit). The **$$REQUEST** command syntax is the same for users of the interactive session access method, but the command is transmitted within the data file, as an XMODEM upload to the local site, along with other $$ commands, if any, and the required /*SIGNON card. **$$REQUEST** cards must precede $$ADD cards.

ACME retrieves all the eligible batches added to the repository at the host site. Eligible batches have requestable flags but do not have a transmitted, nontransmittable, or deleted flag. ACME views these status flags by first issuing a **$$DIRECTORY** command to receive a listing of ACME

batches. Having taken the default parameter, CONV=N, no transmission code conversions are done. ASCII, EBCDIC, and binary batches are retrieved.

ACME collects one or more batches of data as a single, concatenated file because the default batch separation method of BCHSEP=NO was accepted. As each batch in the concatenation is successfully transmitted, Connect:Enterprise immediately flags it as transmitted. Although the requestable flags are retained after transmission, if ACME transmits this same **$$REQUEST** command in the future, those batches are no longer eligible for transmission. Newly added batches, without transmitted flags, are eligible.

Note: If the batch was originally added to the repository with the **Extract/transmit behavior of added batches** parameter on a **transmit repeatedly** setting, it has a multitransmittable flag that prevents it from being flagged as transmitted. This leaves it eligible for transmission.

```
cmd> $$REQUEST ID=ACME PASSWORD=letmein
```

Having obtained batch numbers from a **$$DIRECTORY** report, ACME uses the preceding command to request a specific batch. This command can be used to force retransmission of a previously transmitted batch, overriding the batch transmitted status, or it can be used to select a unique batch. This technique does not permit collection of a deleted batch or a nontransmittable batch, nor does it permit collection of a batch without a requestable flag.

The CONV=A parameter specifies the output as ASCII. If the requested batch is ASCII, it is transmitted unconverted. If it is flagged EBCDIC, it is converted. If it is flagged BINARY, it is not transmitted.

If the batch did not previously have a transmitted flag, it is flagged transmitted after successful transmission and its requestable flag is retained. If the batch was added to the repository with a MULTXMIT=Y parameter, it has a multitransmittable flag that prevents it from being flagged as transmitted and makes it eligible for transmission.

```
cmd> $$REQUEST ID=ACME BID=#775 CONV=A
```

ACME requests a subset of the batches eligible for transmission. By specifying a user batch ID, ACME retrieves only batches with the user batch ID *invoices*.

Because the remote site has been receiving all the batches as a single file, they do not want to dial back into the local site to get the remaining untransmitted batches. To retry the whole transmission and retrieve all the batches from the beginning, ACME concatenates the batches using the **Outbound batch separation** parameter set to **Multiple batches are sent as a single batch...** This value instructs Connect:Enterprise to wait until all eligible batches are successfully transmitted before flagging them as transmitted. If the transmission fails on a subsequent batch in a concatenation, an identical **$$REQUEST** command gets all the batches.

With the **Outbound batch separation** parameter set to **No separation...**, the batches are sent concatenated, but they are flagged as transmitted as each batch is transmitted. When this setting is used, there is a risk of batches being flagged as transmitted early in a transmission that later fails

before all the expected batches are transmitted. Successfully transmitted batches are flagged as transmitted and are no longer available.

```
cmd> $$REQUEST ID=ACME BID='invoices' BCHSEP=OPT3
```

ACME downloads multiple batches with this **$$REQUEST** command, but does not want them concatenated into a single file. BCHSEP=OPT4 instructs Connect:Enterprise to generate a unique file name for every eligible batch transmitted. The file names follow the convention *batchno.dat*, where *batchno* is the 8-digit batch number assigned at the mailbox.

If ACME uses the XMODEM protocol, these file names generated by Connect:Enterprise are not passed in the transmission. With XMODEM downloads, ACME must initiate a download for each file expected, manually specifying a unique file name for each file received. After the first file is received, ACME has less than 30 seconds to initiate another XMODEM download, specifying a new file name for collection of the next batch. The ACME remote user must be ready to initiate subsequent downloads as the collection of each batch finishes.

The ZMODEM, YMODEM, or Kermit protocol passes the local site-generated file names for each batch. After a download is initiated at the remote site, all the batches are collected without manual effort. When the download finishes, the remote site has collected multiple, uniquely named files (*batchno.dat*).

```
cmd> $$REQUEST ID=ACME PASSWORD=letmein BCHSEP=OPT4
```

The remote site uses CONV=E to select batches in EBCDIC or ASCII format and convert ASCII batches to EBCDIC. All data transmitted from the local site is in EBCDIC format. No binary data is collected; only batches flagged ASC or EBC are eligible for transmission.

ACME instructs the local site to limit the number of batches transmitted to the first eligible batch that satisfies the selection criteria (ONEBATCH=Y). The remaining eligible batches must be collected on subsequent connects.

```
cmd> $$REQ ID=ACME PASSWORD=letmein ONEBATCH=Y CONV=E
```

ACME requests the log batch maintained by Connect:Enterprise for ACME (called <<ACTIVITY LOG>>). This batch contains the log of all ADD/REQ operations performed by ACME. This batch is maintained only if Connect:Enterprise has enabled the verification of the data delivery feature for the remote site.

```
cmd> $$REQ RECEIPT
```

The log batch can also be requested by specific batch ID number:

```
cmd> $$REQ ID=RMTID BID=#BNO
```

In this example, the remote user can retrieve the LOGBATCH to verify the receipt of data.

The following command returns all batches created between the **FTIME** and the **TTIME** time stamps and concatenates them into one file called *local filename*.

```
cmd> $$REQ "$$ FTIME=[yy]yymmdd[:hhmm] TTIME=[yy]yymmdd[:hhmm]" local filename
```

Unavailable Batches

Batches with the following flags are not available to the Remote Connect **$$REQUEST** command:

- Nontransmittable
- Deleted
- Transmitted (unless a specific batch number is specified)
- Incomplete (unless a specific batch number is specified)
- Batches lacking the requestable flag are not available to the Remote Connect **$$REQUEST** command. These were collected from a remote site that did not use XMIT=Y, MULTXMIT=Y, or TO=Y on the **$$ADD** card and are not eligible for retransmission.
- Binary batches are not available when CONV=A or CONV=E is specified. Binary batches cannot be converted to ASCII or EBCDIC.
- Batches that do not match the mailbox ID and/or user batch ID are not available.

Chapter 9

Bisync Protocol Commands

This chapter describes the following commands available to remote sites using Connect:Enterprise Bisync communications protocol:

- **$$ADD**—Sends a batch of data to the host site.
- **$$DELETE**—Flags a batch of data as deleted at the host site.
- **$$DIRECTORY**—Requests a formatted listing of batches from the host site.
- **$$REQUEST**—Requests a batch of data from the host site.

This chapter also provides information and instructions to assist you during a Connect:Enterprise session. Before you issue any of these commands, ensure that you meet the prerequisites listed in the following section.

Prerequisites

Obtain the following information from your host site personnel before you issue a Connect:Enterprise command:

- Mailbox ID
- Password and any additional site-specific security
- Phone number and other communications parameters
- Determine the type of data you are sending: binary, ASCII, or EBCDIC

Sending a Command from a Remote Site to Connect:Enterprise

This section describes the procedures that a remote site using Bisync protocol completes to send a command to Connect:Enterprise.

1. Create a file for **$$DIRECTORY, $$REQUEST**, and **$$DELETE** commands or prefix the data file you want to send with a **$$ADD** card, using an editor.

2. Embed the appropriate Connect:Enterprise commands in the file.

> **Note:** Commands are usually placed at the top of the file. However, the **$$ADD** command can be placed anywhere within the batch (as in the case when several files are combined) as long as the **$$ADD** card is on a separate line and the SCAN=Y parameter is specified on the first occurrence of an **$$ADD** command.
>
> If the **$$ADD** cards do not precede the data, the host site administrator must have previously written an exit that can determine the identity of the remote site by the contents of the data records themselves.
>
> **$$REQUEST**, **$$DELETE**, and **$$DIRECTORY** commands must precede **$$ADD** cards and their associated data records.

3. Send the file to the repository using your Bisync communications program. Connect:Enterprise reads the embedded Connect:Enterprise commands and acts accordingly.

> **Note:** If you use a /*SIGNON card, it must be the first record in the data. Otherwise, Connect:Enterprise sets the remote user ID using the information in the first $$ card sent from the remote site.

The commands available for Bisync protocol are described in the following sections.

$$ADD Command

The **$$ADD** command is used to add data to the repository. The **$$ADD** command supports multiple add capabilities (using batch separation), preclassifies data, and sets necessary flags. If you are transmitting both **$$ADD** and **$$REQUEST** cards at the same time, send the **$$REQUEST** cards first.

The permissions are set up for your mailbox by the system administrator in the *mbxacl.conf* file. The contents of this file determine the mailboxes you may add batches to with the **$$ADD** command.

$$ADD Command Format

Optional parameters, listed alphabetically, are in plain font in the following table.

You can use any number of blanks between parameters; however, you cannot use blanks within parameters, except for BATCHID, which can have blanks within its parameters as long as the entire string is enclosed by single quotes.

Command	Parameter
$$ADD	BATCHID='xx...xx'
	CODE=A|E|B
	EO=YES|Y|NO|N
	ID=XXXXXXXX
	INDEX=YES|Y|NO|N\
	MULTXMIT=YES|Y|NO|N
	PASSWORD=XXXXXXXX
	SCAN=YES|Y|NO|N
	TRIGGER=YES|Y|NO|N
	TO=YES|Y|NO|N
	XMIT=YES|Y|NO|N

Optional Parameters

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
BATCHID='xx...xx'	identifies the 1–64 character user batch ID for the batch being added. It must be enclosed in either single or double quotation marks if it contains spaces.
CODE=A|E|B	identifies the data format of the batch being added. The remote user must use this parameter to communicate to the host that the data is non-EBCDIC, using A or B, when necessary. A remote site user can also request that the host site administrator specify **Format of data received from remote site** as **...ASCII...**, in the account definition instead. The remote site is responsible for configuring the 3780 Bisync application to use transparent-mode Bisync protocol when sending non-EBCDIC data. **A** – The batch contains ASCII data. Use transparent-mode Bisync protocol. **E** – The batch contains EBCDIC data. This is the default for remote sites using Bisync protocol. **B** – The batch contains binary data. Use transparent-mode Bisync protocol.
EO=YES|Y|NO|N	**YES|Y** – specifies that after transmission to the host repository, the batch is only allowed to be extracted once and is flagged nontransmittable with an N flag. After local extraction from the host repository, the batch is permanently locked and flagged as nontransmittable and unextractable.

Parameter	Description
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your remote site. If omitted, the ID defaults to the resource account name. To permit remote sites to add batches without using **$$ADD** cards, a host site can create an account definition that lacks the **Password** parameter and that is named the same as the Bisync port or Async device on which the remote connect session is established. If BATCH security is enforced, the ID must be one of the mailbox IDs contained in the **Valid ID list** parameter in the MCD file.
INDEX=YES|Y|NO|N	ensures that record lengths are the same at the time of extraction or retransmission as they are at the time of the add. This is necessary only when record separator characters do not occur within the file being transmitted and there is a need to retain record length information. For example, if a binary file (like an object file from an mainframe site) is being transmitted to Connect:Enterprise in 80-byte blocks with no record separator characters occurring at 80-byte intervals and there is a need to ensure that later it is extracted 80 bytes at a time, specify INDEX=Y. INDEX=Y works with fixed-length and variable-length records. INDEX=N is the default.
MULTXMIT=YES|Y|NO|N	indicates whether the batch is available for multiple transmissions after it is successfully added to the repository. **YES|Y** – Indicates that the batch can be transmitted multiple times. If YES is specified, it overrides the **XMIT** parameter, setting it to YES. When MULTXMIT=Y, the added batch is flagged as multitransmittable at the time it is added to the repository. Unlike with the XMIT=Y parameter, the batch bearing a MULTXMIT=Y parameter is not flagged as transmitted when it is transmitted successfully. This makes it eligible for subsequent transmissions. **NO|N** – Indicates that the batch cannot be transmitted multiple times. This is the default.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise remote command exit for mailbox sites that have enabled custom security. The first $$ card is parsed for a password to match the one specified in the account definition, even when exits are not invoked. If the account password does not match the password supplied by the remote site, the session is disconnected.

Parameter	Description
SCAN = YES|Y|NO|N	controls how Connect:Enterprise processes inbound batches that contain embedded **$$ADD** cards. It is only necessary on the first **$$ADD** card of a batch. **YES|Y** – Embedded **$$ADD** cards cause subsequent data to be added as a separate batch. This is the default. **NO\|N** – Embedded **$$ADD** cards are not processed. They are considered data. When the **Scan for embedded $$ADD cards?** parameter is set to **No...** in the account definition, SCAN=YES **$$ADD** card parameters are honored only on physical batch boundaries. This is also true when a SCAN=NO **$$ADD** card parameter is specified, regardless of the setting of the **Scan for embedded $$ADD cards?** account parameter. After a SCAN=NO **$$ADD** card parameter has been processed, no more embedded **$$ADD** cards are processed until the next physical batch boundary. A physical batch boundary is the first inbound record of a session or the first inbound record following an ETX or EOT. Scanning logic is reset to the value specified in the account definition at each ETX or EOT. **Note**: When SCAN=YES, either through the **Scan for embedded $$ADD cards?** account parameter or the **SCAN $$ADD** card parameter, embedded **$$ADD** cards are scanned (sought). This occurs on every record in a nontransparent mode transfer and on the first record of every block in a transparent mode transfer.
TO=YES|Y|NO|N	**YES|Y** – specifies that the batch can only be transmitted remotely once and cannot be extracted; it is flagged as unextractable with a U flag. After transmission to the intended remote site, the batch is permanently locked and flagged as nontransmittable and unextractable with an N flag and a U flag, respectively. If transmission of a TO=Y batch fails when one or more records have been transmitted, the batch is still locked, but lacks the T flag. To retry the transmission, a new batch must be added from the original source.
TRIGGER=YES|Y|NO|N	allows files to be rerouted immediately to other remote sites. In order for automatic routing to function, an ACD file must be defined at the host site, with the **Contact** parameter set to **Forward data to Remote site automatically** and the **Remote communication sequence (mode)** parameter set to **Send only**. This ACD file must be named with the extension .acd. When the characteristics of the batch being added match the selection criteria in this ACD file, the batch is automatically forwarded to the destination specified in the ACD file. The ID of the **$$ADD** card must match the remote name in the ACD file or the **Send ID(s)** value. **YES|Y** – The batch is rerouted if a valid Auto Connect list with matching selection criteria has been defined. **NO|N** – The batch is not forwarded. This is the default.

Parameter	Description
XMIT=YES|Y|NO|N	indicates whether the batch is available for transmission to another remote site after it is successfully added to the repository. **YES|Y** – Indicates that the batch is transmittable to any remote site that identifies the correct mailbox ID. A requestable flag is set. When XMIT=Y, the added batch is flagged as transmitted after it is successfully sent to another remote site. **NO|N** – Indicates that the batch is not transmittable. The batch can only be extracted at the host site. This is the default.

Invalid $$ADD Parameter Combinations

The following combinations of parameters are invalid:

- Do not use XMIT=Y with MULTXMIT=Y.
- Do not use XMIT=Y with TO=Y.
- Do not use XMIT=Y with EO=Y.
- Do not use TO=Y with MULTXMIT=Y.
- Do not use EO=Y with MULTXMIT=Y.
- Do not use EO=Y with TO=Y.
- Do not use TRIGGER=Y with EO=Y.

Automatic Routing

Refer to the *Auto connect Configuration* chapter of the *CONNECT:Enterprise for UNIX Configuration Files Reference Guide* for a description of Automatic Routing, which allows batches to be automatically forwarded to remote sites as soon as they are added to a mailbox, for example, with a **$$ADD** command.

Examples

A remote site (ACME) transmits a report file to a Connect:Enterprise host site. The data is in EBCDIC format and is identified with a user batch ID of 'sales summary.' The final destination of the data is the host site, so ACME uses the defaults XMIT=N, MULTXMIT=N, TO=N, and because it does not want to limit host site extraction to a single extract, it uses the default EO=N parameter. The card image and the data are transmitted as a single batch in Bisync nontransparent mode. The batch is flagged as collected to indicate collection from a remote site. The batch does not have a requestable flag. It can be extracted by a host site **cmuextract** command. However, without the requestable flag, it cannot be forwarded to another remote site. Upon collection, it is flagged CBK (Collected, Bisync, EBCDIC). After extraction, it is flagged CBKE.

```
$$ADD ID=ACME BID='sales summary'
```

Another EBCDIC file is destined for a single remote site named SMITH after being successfully transmitted to the repository. Using the XMIT=Y parameter causes Connect:Enterprise to put a

requestable flag on the batch. The card image and the data are transmitted as a single batch in Bisync nontransparent mode. Upon collection, the batch is flagged CBKR. If the transmission to the other remote site is unsuccessful, the batch is not flagged as transmitted and remains available for another attempt. A transmitted flag is posted on the batch after it is transmitted successfully and the requestable flag is retained (CBKRT). This transmitted flag makes the batch ineligible for additional transmissions, with one exception: After a transmitted flag is posted, the batch is still available to a remote site that uses a **$$REQUEST** command specifying the batch ID number. The batch ID can be obtained by issuing a **$$DIR** command**.**

Even though ACME makes the batch available for forwarding to another remote site, they allow host site extraction of the batch with **cmuextract** (TO=N by default). The U flag is off. Extraction at the host site posts an extracted flag. However, this flag does not affect the availability of the batch for transmission to another remote site, unless the host site **cmuextract** operation deletes the batch by using the DELETE=YES option. If the batch has been transmitted, extracted, and deleted, the flags are CBKRTED.

In addition, this example shows how ACME uses the ID of SMITH because the remote site SMITH specifies ID=SMITH on the **$$REQUEST** card it sends to collect the available batch. ACME can use the ID=ACME, but the SMITH remote site must specify $$REQUEST ID=ACME. Either way works, but the two remote sites should coordinate the use of mailbox IDs, taking into consideration the security administration at the host site.

```
$$ADD ID=SMITH BID='termination notices' XMIT=Y
```

In the following example, ACME forwards a binary file to multiple remote sites through the Connect:Enterprise host site. The **$$ADD** card and the data are transmitted as a single batch in Bisync transparent mode. The batch is flagged as collected, Bisync, requestable, binary, and multitransmittable (CBRYM). The batch is not flagged as transmitted upon retransmission from the host site mailbox. The multitransmittable and requestable flags remain in effect after transmissions. The only action a remote site can take to make this batch unavailable for transmission is to logically delete it with a **$$DELETE** command. Without delete or transmittable flags, a batch flagged multitransmittable remains permanently eligible for transmission.

```
$$ADD ID=ACME BID='holiday greetings' MULTXMIT=Y CODE=B
```

To prevent host site extraction yet make an ASCII batch available for a single transmission to the remote site SMITH, ACME specifies the parameter TO=Y, as shown in the next example. The unextractable flag indicates transmit once/transmit only status for the added batch. The **$$ADD** card and the data are transmitted as a single batch in Bisync transparent mode. The batch displays the collected, requestable, and unextractable flags when the add finishes. Unextractable batches are permanently locked against host site extraction and after transmission to a remote site, this unextractable batch is flagged as nontransmittable and transmitted (CBRZUNT).

Note: If transmission of this batch fails midway, it is flagged as nontransmittable but not as transmitted, which indicates a failed transmission. Despite the failure, the batch is not retransmitted. After a nontransmittable flag is set, the batch is permanently locked and the originating station, ACME, has to add a duplicate batch to the repository to retry the forwarding transmission.

```
$$ADD ID=SMITH BID='for your eyes only' TO=Y CODE=A
```

The following example illustrates transmitting a report file to the host site. The data is in ASCII format and is identified with the user batch ID 'sales summary'. ACME, the remote site, does not want the batch to be forwarded to other remote sites nor extracted at the host site more than once. The **CODE=** parameter must be used because the data format is not the default, EBCDIC. The card image and the data are transmitted as a single batch in Bisync transparent mode. Connect:Enterprise flags the batch as collected, but not requestable. The EO=Y parameter flags the batch as nontransmittable to reflect the extract-only/extract-once status of the batch. This batch is permanently locked against transmission and upon successful extraction is locked against subsequent extraction with an unextractable flag (CBZNU).

```
$$ADD ID=ACME BID='sales summary' EO=Y CODE=A
```

In the next example, ACME specifies that when the batch used in the preceding example has been received successfully at the host site, Connect:Enterprise forwards it immediately. The system administrator configures the destination to which this batch is forwarded in an ACD file. If the remote site deposits multiple ETX-separated batches in one mailbox ID, it should specify TRIGGER=Y as the last **$$ADD** card only. No unique flags are associated with the use of TRIGGER=Y. After successful transmission, the batch is flagged as transmitted (T).

```
$$ADD ID=ACME BID='Inventory Report' TRIGGER=Y
```

The following example transmits a batch to multiple mailboxes in a single command and generates a multi-add of the file.

```
$$ADD ID=A,B,C BATCHID= 'my_file'
```

$$DELETE Command

This command enables you to mark a specific batch of data as logically deleted from your mailbox using the deleted (D) flag.

The permissions set up for your mailbox by the system administrator in the *mbxacl.conf* file determine the mailboxes from which you can delete batches. Mailbox users can delete batches from their own mailbox. The default is for every mailbox to have access to all mailboxes.

After Connect:Enterprise receives the **$$DELETE** command, it processes the command and returns the results to the remote site. The remote site collects the output to the console.

When a file is logically deleted, a delete flag is added to it, indicating that the file is deleted. At this point, the host site operator can restore the batch using the **cmustatus** command or physically erase it using the **cmuerase** command. If a batch has a D flag, it cannot be transmitted, but it can be extracted at the host site if the batch number is specified with the **cmuextract** command.

$$DELETE Command Format

Optional parameters, listed alphabetically, are in plain font in the following table.

You can use any number of blanks between operands but not within operands. **$$DELETE** can be abbreviated as **$$DEL**.

Command	Parameter
$$DELETE	**BATCHID**='xx...xx'|#nnnnnnnn
	FLAGS=[[!][A|C|D|E|G|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ID=XXXXXXXX
	ORIG=XXXXXXXX
	PASSWORD=XXXXXXXX
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	RECEIPT

The following table describes the available parameters.

Parameter	Description
BATCHID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. The user batch ID can be 1–64 characters. It must be enclosed in either single or double quotation marks if it contains spaces. Wildcard specifications (for example, an asterisk, *) are supported. To specify a batch number, place a pound sign in front of the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.

Parameter	Description
FLAGS=[[!][A|C|D|E|G|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of these flags. You can negate a flag by adding an exclamation point (!) in front of it. For example, A!I specifies to delete all batches added locally with **cmuadd** (A flag), but not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I deletes the batches that are neither added locally nor incomplete. The syntax CM!I specifies to delete batches collected remotely, that are multitransmittable, and not incomplete.
FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for deletion. If the **TTIME** parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **$$DELETE** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **$$DELETE** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=991231:1400 limits the **$$DELETE** command to those batches created on or before December 31, 1999 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm Specifies on or after the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow. The status report indicating which batches were logically deleted is written to standard output. If an error is encountered when the command line is parsed, the error messages are written to standard error.
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your site. The **ID** parameter is optional unless a user batch ID is entered. Wildcard characters (like an asterisk, *) are supported. If you do not specify the ID, it defaults to the logon ID specified in the /*SIGNON.

Parameter	Description
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.
TTIME[CC]yymmdd[:hhmm]| nnn[:hhmm]]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which or before which a batch must have been created to be eligible for deletion. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **$$DELETE** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **$$DELETE** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **$$DELETE** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another method is to use the hhmm subparameter without the colon (:) character. The entry TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for deletion. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm Specifies on or before the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise remote command exit for those mailbox sites that have enabled custom security. The value supplied is used when exits are not enabled. If a password is defined in the account definition, the inbound password must match it.
RECEIPT	deletes the log batch called the <<ACTIVITY LOG>>. It is a log of all the ADD/REQ activity performed by the remote site. By doing this, it resets the batch. This parameter cannot be used with any other parameter.

Examples

ACME deletes a specific batch by specifying the batch number. The ID and PASSWORD are optional for processing the **$$DELETE** command, but are required by host site security, as in this example. This command logically deletes batch number 775, posting a delete flag. Deleted batches are not available for transmission, even if the requestable flag is on and a specific batch number is

specified on the **$$REQUEST**. Deleted batches can be extracted at the host site, if the specific batch number is specified with the **cmuextract** command. Batches can be restored only at the host site with the **cmustatus** command. The batch is unrecoverable, within the scope of Connect:Enterprise, after a host site **cmuerase** command physically erases it.

```
$$DELETE ID=ACME BATCHID=#775 PASSWORD=letmein
```

The following command turns on the delete flag for every batch with the mailbox ID ACME.

```
$$DELETE ID=ACME PASSWORD=letmein
```

ACME narrows the deletion of ACME batches to those with a user batch ID of ’invoices’, and the mailbox ID ACME.

```
$$DELETE ID=ACME BATCHID=’invoices’ PASSWORD=letmein
```

ACME deletes the batch maintained by Connect:Enterprise for ACME (called the <<ACTIVITY LOG>>).

```
$$DEL RECEIPT
```

After the log batch is erased, the next add or extract of batches creates a new LOGBATCH for the remote user.

$$DIRECTORY Command

The **$$DIRECTORY** command displays the contents of your mailbox and batch status information.

The permissions for your mailbox are set up by the system administrator in the *mbxacl.conf* file. These permissions determine the mailboxes for which users can obtain directory listings. If you enter a **dir** command for a mailbox to which you do not have access, you get the message *Permission denied.*

Batch selection is by mailbox ID alone, mailbox ID and user batch ID, by batch number, originator, flags, and start and end times.

This command does not display batches with a D (deleted) flag. The contents are displayed either to standard output or to a local file if one is specified as an argument.

$$DIRECTORY Command Format

The required parameter is in bold in the following table.

You can use any number of blanks between operands but not within operands, except for the BATCHID parameter. **$$DIRECTORY** can be abbreviated to **$$DIR.**

Command	Parameter
$$DIRECTORY	**ID=XXXXXXXX**
	BATCHID='xx...xx'|#nnnnnnnn
	BLOCK=nnn
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ORIG=XXXXXXXX
	PASSWORD=XXXXXXXX
	RECEIPT
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

Required Parameter

The following parameter table describes the available parameters. Required parameters are in bold.

Parameter	Description
ID=xxxxxxxx	identifies the 1–8 character mailbox ID for which directory information is desired. Connect:Enterprise searches the mailbox for all batches that match the specified mailbox ID. Wildcard characters (like an asterisk, *) are supported.

Parameter	Description
BATCHID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. The user batch ID can be 1–64 characters. It must be enclosed in either single or double quotation marks if it contains spaces. Wildcard specifications (like an asterisk, *) are allowed. To specify a batch number, place a pound sign before the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign, and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.
BLOCK=nnn	overrides the **Block** parameter in the account definition, where nnn specifies the number of records. If number is not specified, the transmit buffer is filled with as many logical records as can fit.
FLAGS=[[!][A|C|D|E|G|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of these flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I specifies to obtain a directory listing for all batches added locally with **cmuadd** (A flag), but not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I requests a directory listing of the batches that are neither added locally nor incomplete. The syntax CM!I specifies a directory request for batches collected remotely, that are multitransmittable, and not incomplete.

Parameter	Description
FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for reporting. If the TTIME parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **$$DIRECTORY** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **$$DIRECTORY** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. The entry FTIME=020101:1315 limits the **$$DIRECTORY** command to those batches created on or after January 1, 2001 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for reporting. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm Specifies on or after the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the mailbox.
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise remote command exit for those mailbox sites that have enabled custom security. The first $$ card is parsed for a password to match the one specified in the account definition, even when exits are not invoked. If the account definition contains a password and that value has not been supplied by the remote site, the session is disconnected.
RECEIPT	causes Connect:Enterprise to send a directory listing of the log batch called the <<ACTIVITY LOG>>. It is a log of all the ADD/REQ activity done so far by the remote. This batch can be considered as a receipt by the remote site. This batch is reset by the command **$$DELETE** RECEIPT. This parameter cannot be used with any other parameter.

Parameter	Description
TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which or before which a batch must have been created to be eligible for deletion. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **$$DIRECTORY** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **$$DIRECTORY** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **$$DIRECTORY** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm Specifies on or before the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

Output

After a **$$DIR** command is issued, Connect:Enterprise scans the mailbox for the designated batches and returns that information to the remote site in the following format:

```
xxxxxxxx nnnnnnnn bbbbbbbbbb <x...x> YYMMDD-HHMM flags protocol format
```

The following table describes the output:

Section	Description
xxxxxxxx	The 8-character mailbox ID for the batch.
nnnnnnnn	The 8-digit batch number assigned to the batch.

Section	Description
bbbbbbbbbb	The total byte count of the batch.
<x...x>	The first 24 characters of the user batch ID. If the user batch ID is longer than 24 characters (up to the 64-byte maximum), it is shortened to the first 23 characters of the literal with the 24th character becoming a plus sign (+).
YYMMDD-HHMM	The date and time that the batch was added to the repository.
flags	One or more flags identifying the current status of a batch in the repository. Refer to *Flags* on page 11 for a complete list of all flags available. For information concerning the **cmustatus** command, refer to the *Local User Commands* chapter in the *CONNECT:Enterprise for UNIX User's Guide.*
protocol	indicates the transmission protocol used to place the batch into the specified mailbox. The following options are possible: ◆ TCP Transmitted locally with **cmuadd** command ◆ ASY Transmitted remotely with Async protocol ◆ BSC Transmitted remotely with Bisync protocol ◆ FTP Transmitted remotely with FTP protocol ◆ FTS Transmitted remotely with Secure FTP protocol ◆ (blank) Activity log batch
format	indicates the data format of the batch. Valid values are ASC (ASCII), BIN (binary), or EBC (EBCDIC).

Examples

The following example illustrates listing all batches stored in the mailbox with ID=ACME.

```
$$DIR ID=ACME
```

In the next example, ACME improves throughput by sending the batch information records blocked 30 records to a Bisync message block. The default record separator of x'1E' is used. Only batches with the user batch ID of 'orders' are selected. When a user batch ID is specified, a mailbox ID is required. All records are returned to ACME as a single, concatenated batch because BCHSEP=NO is the default.

```
$$DIR ID=ACME BID='orders' BLOCK=30
```

The next example show how to view the characteristics and status of a specific batch number obtained previously in another operation.

```
$$DIR ID=ACME BID=#787
```

ACME does not want the returned records to be concatenated into a single batch. BCHSEP=OPT2 instructs Connect:Enterprise to separate each batch information record into discrete batches using

ETX batch separation. OPT1 causes batch separation also, but with the common RJE method using ETX/EOT sequences.

```
$$DIR ID=ACME BID='payable' BCHSEP=OPT2
```

In the following example, ACME requests directory information for the batch maintained by Connect:Enterprise for ACME. This batch contains the log of all ADD/REQ operations performed by ACME. This batch is maintained only if Connect:Enterprise has enabled the verification of data delivery feature for the remote site.

After the log batch is erased, the next add or extract of batches creates a new LOGBATCH for the remote user.

```
$$DIR ID=REMOTE BID='activity log'
```

In the following example, the **$$DIR** command returns all batches whose originator ID is the *originator name* indicated (within the constraints set forth by the ACL defaults section). In other words, if the current directory has no batches available to the current user (based on that user's permissions), then the **DIR** command does not return any batches.

```
$$DIR "$$ ORIG=originator name"
```

The following example gives a directory listing of all batches originated by Dave at or after July 22, 2002, 12:30.

```
$$DIR "$$ORIG=dave FROM=20020722:1230"
```

The **FLAGS** parameter is also a filtering identifier.

```
$$DIR "$$ [FLAGS=char1[char2...]]"
```

The following example returns all batches that are flagged for multiple transmission (M) and are requestable (R).

```
$$DIR "$$ FLAGS=MR"
```

$$REQUEST Command

Data transmission from the host site is initiated by sending a **$$REQUEST** command. The **$$REQUEST** command can specify a single batch number, all batches for the specified mailbox ID, or batches that have a specific mailbox ID and match a specified user batch ID (alphanumeric string). In the latter case, the BATCHID may contain wildcard specifications to further define which batches the remote site wants.

The permissions set up for your mailbox by the system administrator in the *mbxacl.conf* file determine where you can request batches from. Every mailbox is allowed to request batches from their own mailbox. The default is for every mailbox to have access to all mailboxes.

When multiple batches are requested (or the potential for multiple batch transmission exists), only batches that are not already marked as transmitted are selected. When a specific batch is requested by batch number, it is transmitted, regardless of its transmission status. Further restrictions can be placed on the batches transmitted to provide automatic character set translation (CONV=option).

Note: If you are transmitting both **$$ADD** and **$$REQUEST** cards at the same time, send the **$$REQUEST** cards first.

The following rules apply when truncation and compression are specified:

- When truncation and compression are requested, trailing blanks are truncated before compression is performed on embedded blanks.
- Bisync transparency mode cannot be used in conjunction with blocked data, truncation, or compression.

$$REQUEST Command Format

The required parameters are in bold in the following table. Optional parameters are in plain font.

You can use any number of blanks between operands, but not within operands, except the BATCHID operand. The **$$REQUEST** command can be abbreviated to **$$REQ**.

Command	Parameter
$$REQUEST	**ID=XXXXXXXX**
	BATCHID='xx...xx'|#nnnnnnnn
	BCHSEP=NO|OPT1|OPT2|OPT3
	BLOCK=nnn
	CMP=Y|N
	CONV=A|E|N
	FLAGS=[[!][A|C|D|E|I|M|R|T|B|F|G|Q|Y|Z|K]]...
	FTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]
	ONEBATCH=YES|Y|NO|N
	ORIG=XXXXXXXX
	PASSWORD=XXXXXXXX
	RECEIPT

Command	Parameter
	TRUNC=Y|N
	TTIME=[CC]yymmdd[:hhmm]|nnn[:hhmm]]

Required Parameters

The following table describes the available parameters. Required parameters are in bold.

Parameter	Description
ID=xxxxxxxx	identifies the 1–8 character mailbox ID assigned to your site. If the batch level security is turned on in the Connect:Enterprise MCD file using the **Remote Batch Security** parameter, the ID must be one of the IDs contained in the MCD **Valid Mailbox ID list** parameter. Wildcard characters (like an asterisk, *) are allowed.
BATCHID='xx..xx' | #nnnnnnnn	identifies either the user batch ID or the batch number. The user batch ID can be 1–64 characters. It must be enclosed in either single or double quotation marks if it contains spaces. Wildcard specifications (like an asterisk, *) are allowed. To specify a batch number, place a pound sign in front of the batch number (for example, #14). The batch number can be up to eight digits. Leading zeros are not required. One or more hyphenated ranges of batch numbers can be specified after the pound sign and these can be separated by commas. Ranges can be mixed with unique batch numbers (for example, #57–59,88,95,100–110,128). The string must not be longer than 64 characters including the pound sign.

Parameter	Description
BCHSEP=NONE|OPT1|OPT2|OPT3	specifies the batch separation method that Connect:Enterprise uses to send data to the remote site. Options are: **NONE** – Specifies that batches are not separated. If multiple batches are sent in a single connection, they are concatenated and sent as a single batch. Batches are flagged as transmitted throughout the transmission, as each batch is successfully transmitted. **OPT1** – Specifies that batches are separated using the common RJE method of separating data. Each batch ends with an ETX-terminated message block, as in OPT2, but upon receipt of an ACK-n, EOT is sent to the remote site. Connect:Enterprise reads a response from the remote site, and sends ENQ to request use of the line to send the next batch. **OPT2** – Specifies that batches are separated with ETX-terminated message blocks. **OPT3** – Specifies that batches are not separated. If multiple batches are sent in a single connection, they are concatenated and sent as a single batch, like BCHSEP=NO, but batches are not marked as transmitted until all the batches matching the **$$REQUEST** criteria have been transmitted successfully. Connect:Enterprise obtains default values from the account definition, but these can be overridden.
BLOCK=nnn	overrides the **Block** parameter in the account definition, where nnn specifies the number of records. If number is not specified, the transmit buffer is filled with as many logical records as can fit.
CMP=YES|Y|NO|N	identifies whether Bisync 3780 blank compression is performed on the data before transmission. During transmission, Bisync compression reduces strings of three or more bytes of blanks (40) to a compression indicator byte (1D) and a length byte to indicate the number of blanks represented by the two bytes. A receiving station, on detecting the 1D compression indicator, reads the length byte that follows and decompresses the two-byte string to the original string of blanks. Do not request compression for transparent batches. When blank compression and blank truncation are both enabled, truncation of trailing blanks is performed first, followed by space compression of embedded strings of three or more blanks. **YES|Y** – Specifies that Bisync 3780 blank compression is performed before data transmission. This option is only allowed for EBCDIC data for transmission in nontransparent blocks. **NO|N** – Specifies that embedded repeating blanks are transmitted to the remote site. This is the default.

Parameter	Description
CONV=A|E|N	specifies the format for data transmitted to the remote site. Because some conversions are not allowed (for example, binary to ASCII), this parameter can limit the batches selected for transmission. The **CONV** parameter also determines the type of record conversion Connect:Enterprise performs, if any. The CONV processing rules can be specified in the account definition using the **Outbound data format** parameter, but those can be overridden. The default is CONV=N. **A** – Specifies to transmit ASCII data to the remote site. When A is specified, batches flagged as containing ASCII (Z) or EBCDIC (K) data and that meet the **$$REQ** requirements are transmitted to the remote site. EBCDIC batches that are selected are translated to ASCII based on user-supplied or Sterling Commerce-supplied translation tables (specified in the account definition or a remote block of a schedule). No binary batches can be transmitted. Bisync transparent-mode message blocks are used. **E** – Specifies to transmit EBCDIC data to the remote site. When E is specified, batches marked as containing ASCII or EBCDIC data and meeting the **$$REQ** requirements are transmitted to the remote site. ASCII records are translated to EBCDIC through user-supplied or Sterling Commerce-supplied translation tables. No binary batches can be transmitted. Non-transparent Bisync message blocks are used. **N** – Specifies that no translation is to occur. All batches that meet the **$$REQ** requirements are transmitted to the remote site without translation. Transparent mode batches are mixed with nontransparent mode batches. If this is unacceptable to the Bisync 3780 emulation application of the remote site, some other selection criteria must be invoked to segregate EBCDIC batches from those that are sent with transparent mode. This is the default.
FLAGS=[[!][A|C|D|E|G|I|M|R|T|B|F|G|Q|Y|Z|K]]...	filter by specific process flags. Refer to *Flags* on page 11 for definitions of these flags. You can negate a flag by adding an exclamation point (!) in front of the flag. For example, A!I specifies to get all batches added locally with **cmuadd** (A flag), but not marked incomplete (!I). These flags can be used in conjunction with other flags to create an *and* scenario. For example, the syntax !A!I specifies to get the batches that are neither added locally nor incomplete. The syntax CM!I requests batches collected remotely, that are multitransmittable, and not incomplete.

Parameter	Description
FTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]]	specifies the earliest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which a batch could have been created to be eligible for reporting. If the TTIME parameter is omitted, the current date and time are assumed as the period end. Specifying FTIME=020101 limits the **$$REQ** command to those batches created on or after January 1, 2002. Specifying FTIME=25 also limits the **$$REQ** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. Specifying FTIME=20011231 with TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameters. The entry FTIME=020101:1315 limits the **$$REQ** command to those batches created on or after January 1, 2002 at 1:15 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for reporting. Another option is to use the hhmm subparameter without the colon (:) character. FTIME=1315 specifies that all batches created after 1:15 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ FTIME=[CC]yymmdd Specifies on or after the date [CC]yymmdd ◆ FTIME=[CC]yymmdd:hhmm Specifies on or after the date and time [CC]yymmdd and hhmm ◆ FTIME=nnn Specifies on or after the date nnn days ago ◆ FTIME=nnn:hhmm Specifies on or after the date and time nnn days ago and hhmm ◆ FTIME=hhmm Specifies on or after the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.
ONEBATCH=YES|Y|NO|N	indicates that only the first batch matching the selection criteria is transmitted to the remote site in response to the **$$REQ** command. **YES|Y** – When BID= is specified, this parameter indicates that only the first matching batch is returned. **NO|N** – Specifies that all batches that conform to the request are to be transmitted to the remote site. This is the default.
ORIG=xxxxxxxx	specifies the originator of the batch. The originator is the user who originally put the batch in the repository.

Parameter	Description
PASSWORD=xxxxxxxx	enables a remote site to supply a password to the Connect:Enterprise remote command exit for those mailbox sites that have enabled custom security. The first $$ card is parsed for a password to match the one specified in the account definition, even when exits are not invoked. If the account definition contains a password and that value has not been supplied by the remote site, the session is disconnected.
RECEIPT	causes Connect:Enterprise to send a log batch called the <<ACTIVITY LOG>>. It is a log of all the ADD/REQ activity performed by the remote. This batch can be considered as a receipt by the remote site. This batch is reset by the command **$$DELETE** RECEIPT. This parameter cannot be used with any other parameter.
TTIME=[CC]yymmdd[:hhmm] |nnn[:hhmm]]	specifies the latest date ([CC]yymmdd), or number of days (nnn) prior to the current date, on which or before which a batch must have been created to be eligible for deletion. If the **FTIME** parameter is omitted, the creation date and time of the oldest batch are assumed as the period start. Specifying TTIME=011231 limits the **$$REQUEST** command to those batches created on or before December 31, 2001. Specifying TTIME=25 also limits the **$$REQUEST** command to batches created on or before December 31, 2001, assuming today's date is January 25, 2002. Two-digit and four-digit year formats are supported. If [CC]=20 and yy=01, the year is 2001. The entry FTIME=20011231 TTIME=20020101 lists all batches created on or between December 31, 2001 and January 1, 2002. This parameter can be further defined using the :hhmm subparameter. Specifying TTIME=011231:1400 limits the **$$REQUEST** command to those batches created on or before December 31, 2001 at 2:00 pm. The :hhmm subparameter requires either [CC]yymmdd or nnn. Another option is to use the hhmm subparameter without the colon (:) character. The entry TTIME=1400 specifies that all batches created on or before 2:00 pm today are eligible for retrieval. The possible valid combinations are as follows: ◆ TTIME=[CC]yymmdd Specifies on or before the date [CC]yymmdd ◆ TTIME=[CC]yymmdd:hhmm Specifies on or before the date and time [CC]yymmdd and hhmm ◆ TTIME=nnn Specifies on or before the date nnn days ago ◆ TTIME=nnn:hhmm Specifies on or before the date and time nnn days ago and hhmm ◆ TTIME=hhmm Specifies on or before the time hhmm today **Note**: Do not use the **TTIME** parameter without using the **FTIME** parameter. Although it is valid to omit the **FTIME** parameter, the generation of a default date and time for the start of the search is very slow.

Parameter	Description
TRUNC=YES|Y|NO|N	specifies whether Connect:Enterprise can truncate trailing blanks before records are transmitted. **YES|Y** – Specifies that trailing blanks are truncated prior to data transmission. This option is only allowed for EBCDIC data, for transmission in nontransparent blocks. **NO|N** – Specifies that trailing blanks are transmitted to the remote site. This is the default.

Examples

ACME wants to retrieve all the eligible batches that have been added to the repository at the host site. Eligible batches are batches that have requestable flags but are not flagged as transmitted, nontransmittable, or deleted. ACME can view the status flags by first transmitting a **$$DIRECTORY** command to receive a listing of ACME's batches. A host site administrator has provided ACME with a password to be included on the **$$REQUEST** command.

ACME initiates the connection (a Remote Connect) and transmits the **$$REQUEST** card, then enters Receive Mode to collect one or more batches of data as a single, concatenated batch because the default batch separation method is BCHSEP=NO. As each batch in the concatenation is successfully transmitted, Connect:Enterprise immediately flags it as transmitted. Although the requestable flags are retained after transmission, if ACME transmits this same **$$REQUEST** in the future, those batches are no longer eligible. However, newly added batches, lacking transmitted flags, are eligible.

Note: If the batch is originally added to the repository with a MULTXMIT=Y parameter, it has a multitransmittable flag that prevents it from being flagged as transmitted. This leaves it eligible for transmission to other sites or to the same site multiple times.

```
$$REQUEST ID=ACME PASSWORD=letmein
```

To request a specific batch, having previously obtained batch numbers from a **$$DIRECTORY** report, ACME issues the preceding command. This command can be used to force retransmission of a previously transmitted batch or to select a unique batch. This does not permit collection of a deleted batch or a nontransmittable batch, nor does it permit collection of a batch that lacks a requestable flag. If the batch does not previously have a transmitted flag, it is flagged transmitted after successful transmission and its requestable flag is retained.

Note: If the batch is added to the repository with a MULTXMIT=Y parameter, it has a multitransmittable flag that prevents the batch from being flagged as transmitted. This leaves it eligible for transmission to other sites or to the same site multiple times.

```
$$REQUEST ID=ACME BID=#775
```

To request a subset of all the batches that are eligible for transmission, ACME specifies the user batch ID used when the batches were added to the repository; therefore, ACME retrieves only batches with the user batch ID of *invoices*.

Because the remote site, ACME, has been receiving all the batches as a single file, they don't want to dial back into the host site get the remaining intransitive batches. ACME retries the whole transmission and retrieves all the batches from the beginning. To avoid this problem, ACME concatenates the batches using BCHSEP=OPT3. This instructs Connect:Enterprise to wait until all eligible batches are successfully transmitted before flagging them all as transmitted. If the transmission fails on the second or subsequent batch in a concatenation, a subsequent, identical **$$REQUEST** command gets all the batches.

With BCHSEP=NO, the batches are sent concatenated, but they are flagged as transmitted as each batch is transmitted. When BCHSEP=NO is used, there is a risk of batches being flagged as transmitted early in a transmission that later fails before all the expected batches are transmitted. Successfully transmitted batches are flagged as transmitted and are no longer available.

```
$$REQUEST ID=ACME BID='invoices' BCHSEP=OPT3
```

CONV=E limits selection of batches to EBCDIC and ASCII format and converts those in ASCII to EBCDIC. All data transmitted from the host site is in EBCDIC format. No binary data is collected.

In the following example, ACME uses ONEBATCH=Y to transmit the first eligible batch that satisfies the selection criteria of ID=ACME and CONV=E. Any remaining eligible batches are collected on subsequent connects.

```
$$REQ ID=ACME PASSWORD=letmein ONEBATCH=Y CONV=E
```

ACME requests the batch maintained by Connect:Enterprise for ACME. This batch contains the log of all ADD/REQ operations performed by ACME. This batch is maintained only if Connect:Enterprise has enabled the verification of data delivery feature for the remote site.

```
$$REQ RECEIPT
```

Or by naming the specific batch ID number:

```
$$REQ ID=RMTID BID=#BNO
```

In this example, the remote user can retrieve the log batch to verify the receipt of data.

The following command returns all batches created between the FTIME and the TTIME time stamps and concatenates them into one file called *local filename*.

```
$$REQ "$$ FTIME=[yy]yymmdd[:hhmm] TTIME=[yy]yymmdd[:hhmm]" local filename
```

Unavailable Batches

Batches are unavailable in the following circumstances:

- Batches flagged as nontransmittable are not available to Remote Connect **$$REQUEST** or **get** (FTP) command.
- Batches flagged as deleted are not available to the Remote Connect **$$REQUEST** or **get** (FTP) command.
- Batches lacking the requestable flag are not available to the Remote Connect **$$REQUEST** or **get** (FTP) command. The remote site did not use XMIT=Y, MULTXMIT=Y, or TO=Y on the **$$ADD** card and the files are not eligible for retransmission.
- Batches flagged as transmitted are not available to the Remote Connect **$$REQUEST** or **get** (FTP) command unless a specific batch number is specified.
- Batches flagged as incomplete are not available to the Remote Connect **$$REQUEST** or **get** (FTP) command unless a specific batch number is specified.
- Binary batches are not available when CONV=A or CONV=E are specified. Binary batches cannot be converted to ASCII or EBCDIC.
- Batches that do not match the mailbox ID and/or user batch ID are not available.

Index

A

Async 10
  requirements 129
  transmission protocol 133

automatic routing
  $$ FTP 79, 108

B

batch
  inbound flags
    $$ syntax 78
  unavailable
    $$ syntax 66, 104
    standard FTP syntax 28, 85

Bisync commands 159

C

cd
  examples 19, 39, 83, 86, 87, 88, 89
  format 18, 38, 83, 85, 88
  instructions 18, 38, 83, 85, 86, 88

changing your password 13
  remote using $$ FTP 45
  remote using standard FTP 17

Commands 159, 160, 170, 176
  $$ syntax
    del 46, 91
    dir 50, 93
    get 58, 98
    mget 66
    prerequisites 43
    put 73, 104
    recv 58, 98
  standard FTP syntax
    cd 18, 38, 83, 85, 86, 88
    del 20
    dir 22
    get 26, 84
    mdel 28
    mget 31
    mput 34
    prerequisites 15, 81
    put 36
    pwd 38, 87
    recv 26, 84

communication lines 10

communication parameters 133

D

data bits
  communication parameters 133

data format flags 11
  inbound batches
    $$ syntax 78

del
  $$ syntax
    examples 50, 93
    format 46, 92
    instructions 46, 91
  standard FTP syntax
    examples 21
    format 21
    instructions 20

dir
  $$ syntax
    examples 55, 97
    format 51, 93
    instructions 50, 93
    output 55, 96
  standard FTP syntax
    examples 24
    format 22, 26, 84
    instructions 22
    output 23

E

end a session 10

establish a connection 10

F

filtering
  enhanced
    $$ syntax 57, 98
    async 149

flags 11
  inbound batches
    $$ syntax 78

FTP $$ Commands 43, 91

FTP Commands
  $$ syntax
    del 46, 91
    dir 50, 93
    get 58, 98
    mget 66
    put 73, 104
    recv 58, 98
    remote site script 79
    requirements 43
  standard FTP syntax
    cd 18, 38, 83, 85, 88
    del 20
    dir 22
    get 26, 84
    logging on 16, 44, 82
    mdel 28
    mget 31
    mput 34
    put 36
    pwd 38, 87
    recv 26, 84
    requirements 15, 81

G

get
  $$ syntax
    examples 62, 103
    format 58, 99, 100
    instructions 58, 98
  standard FTP syntax
    examples 27, 84
    format 26, 84, 85
    instructions 26, 84

H

host-initiated communications 11

I

interactive session
  access method 130
  logging in 132

K

Kermit 129

L

leased line
  definition 10

logging on 10
  remote using $$ FTP 44
  remote using standard FTP 16, 82
  security 10

M

mdel
  examples 29
  instructions 28

mget
  $$ syntax
    examples 70
    format 67
    instructions 66
  standard FTP syntax
    examples 32
    format 31
    instructions 31

mput
  examples 34
  format 34
  instructions 34

multi-add
  async 138
  FTP $$ syntax 78, 107

N

noninteractive access method 130

noninteractive remotes
  prefix data with $$ cards 132

O

ORIG example
  $$ syntax 57, 98
  async 149

P

parity
  communication parameters 133

process flags 11

protocol flags 11

put
  $$ syntax
    examples 75, 106
    format 73, 104
    instructions 73, 104
    invalid parameters 78, 107
    required parameters 74
  standard FTP syntax
    examples 37
    format 36, 86
    instructions 36

pwd
  examples 38, 87
  format 38, 87
  instructions 38, 87

Q

quit
  standard FTP syntax 18, 46, 82, 91

R

recv
  $$ syntax 58, 98
    format 58, 99
  standard FTP syntax
    examples 27, 84
    instructions 26, 84

remote site script
  FTP $$ commands 79

remote user
  communication parameters 133

rename 38

S

SCP 109

script
  remote FTP
    $$ syntax 79

Secure Copy 109

security
  customer-supplied 10
  logging on 10
  sending and receiving data 10
  system 9

sending and receiving data
  security 10

SPC
  noninteractive remotes 132

start a session 10

stop bits
  communication parameters 133

switched line
  definition 10

syntax
  wildcard 13

system security 9

U

unavailable batches
  FTP $$ syntax 66, 104
  standard FTP syntax 28, 85

W

WebDAV 119

wildcard syntax 13

X

XMODEM 129

Y

YMODEM 129

Z

ZMODEM 129
  logging off 130

As set forth below, certain of the Third Party Licensors assert the following terms with respect to their respective products. Such terms shall only apply as to the specific Third Party Licensor product and not to those portions of the product derived from other Third Party Licensor products or to the Sterling Commerce Software product as a whole.

Those portions of the Sterling Commerce Software which include, or are distributed on the same storage media with, the Third Party Software where use, duplication, or disclosure by the United States government or a government contractor or subcontractor, are provided with RESTRICTED RIGHTS under Title 48 CFR 2.101, 12.212, 52.227-19, 227.7201 through 227.7202-4, FAR 52.227-14 and FAR 52.227-19(c)(1-2) and (6/87), and where applicable, the customary Sterling Commerce license, as described in Title 48 CFR 227-7202 with respect to commercial software and commercial software documentation including DFAR 252.227-7013, DFAR 252.227-7014, DFAR 252.227-7015 and DFAR 252.227-7018, all as applicable.

The Sterling Commerce Software and the related documentation are licensed either "AS IS" or with a limited warranty, as set forth in the Sterling Commerce license agreement. Other than any limited warranties provided, NO OTHER WARRANTY IS EXPRESSED AND NONE SHALL BE IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE OR FOR A PARTICULAR PURPOSE. The applicable Sterling Commerce entity reserves the right to revise this publication from time to time and to make changes in the content hereof without the obligation to notify any person or entity of such revisions or changes.

Except as otherwise set forth below, the Third Party Software is provided 'AS IS' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. FURTHER, IF YOU ARE LOCATED OR ACCESSING THIS SOFTWARE IN THE UNITED STATES, ANY EXPRESS OR IMPLIED WARRANTY REGARDING TITLE OR NON-INFRINGEMENT ARE DISCLAIMED.

XALAN-J AND XERCES-J

The Sterling Commerce Software is distributed with or on the same storage media as the Xalan-J and Xerces-J software (together, the "Xalan and Xerxes Software") located at $CMUHOME/javalib/xalan.jar, xerxesImpl-2_1_1.jar, and xmlParserAPIs-2_1_1.jar. Use of the Xalan and Xerxes Software is subject to the terms of the following license:

The Apache Software License, Version 1.1

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software
Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names "Xalan", "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JETTY SOFTWARE
The Sterling Commerce Software is distributed with or on the same storage media as the Jetty Software, located at $CMUHOME/jetty-4.2.21-jdk1.2.jar, which is subject to the following license:

From http://jetty.mortbay.org/jetty/LICENSE.html:

Jetty License
$Revision: 3.7$

Preamble:
The intent of this document is to state the conditions under which the Jetty Package may be copied, such that the Copyright Holder maintains some semblance of control over the development of the package, while giving the users of the package the right to use,

distribute and make reasonable modifications to the Package in accordance with the goals and ideals of the Open Source concept as described at http://www.opensource.org.

It is the intent of this license to allow commercial usage of the Jetty Package, so long as the source code is distributed or suitable visible credit given or other arrangements made with the copyright holders.

Definitions:

- "Jetty" refers to the collection of Java classes that are distributed as a HTTP server with servlet capabilities and associated utilities.
- "Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.
- "Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder.
- "Copyright Holder" is whoever is named in the copyright or copyrights for the package. Mort Bay Consulting Pty. Ltd. (Australia) is the "Copyright Holder" for the Jetty Package.
- "You" is you, if you're thinking about copying or distributing this Package.
- "Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)
- "Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

0. The Jetty Package is Copyright © Mort Bay Consulting Pty. Ltd. (Australia) and others. Individual files in this Package may contain additional copyright notices. The javax.serviet packages are copyright
Sun Microsystems Inc.
1. The Standard Version of the Jetty package is available from
http://jetty.mortbay.org.
2. You may make and distribute verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you include this license and all of the original copyright notices
and associated disclaimers.
3. You may make and distribute verbatim copies of the compiled form of the Standard Version of this Package without restriction, provided that you include this license.
4. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.
5. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do
at least ONE of the following:

a) Place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as ftp.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.
b) Use the modified Package only within your corporation or organization.
c) Rename any non-standard classes so the names do not conflict with standard classes, which must also be provided, and provide a separate manual page for each non-standard class that clearly documents how it differs from the Standard Version.
d) Make other arrangements with the Copyright Holder.
6. You may distribute modifications or subsets of this Package in source code or compiled form, provided that you do at least ONE of the following:
a) Distribute this license and all original copyright messages, together with instructions (in the about dialog, manual page or equivalent) on where to get the complete Standard Version.
b) Accompany the distribution with the machine-readable source of the Package with your modifications. The modified Package must include this license and all of the original copyright notices and associated disclaimers, together with instructions on where to get the complete Standard Version.
c) Make other arrangements with the Copyright Holder.
7. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you meet the other distribution requirements of this license.
8. Input to or the output produced from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whomever generated them, and may be sold commercially,
and may be aggregated with this Package.
9. Any program subroutines supplied by you and linked into this Package shall not be considered part of this Package.
10. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.
11. This license may change with each release of a Standard Version of the Package. You may choose to use the license associated with version you are using or the license of the latest Standard Version.
12. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
13. If any superior law implies a warranty, the sole remedy under such shall be, at the Copyright Holders option either a) return of any price paid or b) use or reasonable endeavors to repair or replace the software.
14. This license shall be read under the laws of Australia.
The End

This license was derived from the Artistic license published on
http://www.opensource.com

OPEN SSL SOFTWARE
The Sterling Commerce Software is distributed with or on the same storage media as the OpenSSL Software, located at$CMUHOME/<os>/bin/*ftp*. The OpenSSL Software is distributed under a dual license, as set out below:

OpenSSL License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: “This product includes software developed by the OpenSSL Project for use in
the OpenSSL Toolkit. (http://www.openssl.org/)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment:
"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS"' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).
This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com).

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com). Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young
(eay@cryptsoft.com)" The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related.
4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF

THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. The license and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License].

OPENSSH SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the OpenSSH Software, located at $CMUHOME/<os>/bin/*ssh*. Copyright © 1995 Tatu Ylonen <ylo@cs.hut.fi>. Any information and cryptographic algorithms used in the OpenSSH Software are publicly available on the Internet and more information can be found at http://www.cs.hut.fi/crypto.

ANTLR

The Sterling Commerce software is distributed with or on the same storage media as the Antlr software, located at $CMUHOME/javalib/antlr.jar, which is subject to the following license:

[The BSD License]
Copyright (c) 2005, Terence Parr
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

COMMONS CODE C, FOP, COMMONS BEANUTILS PACKAGE, COMMONS LOGGING PACKAGE, HIVEMIND, TEXT ORO, XALAN-JAVA, AVALON, BATIK, CATALINA, TOMCAT, NAMING-RESOURCES,, LOG4J AND HTTP CLIENT.
The Sterling Commerce Software is distributed with or on the same storage media as the following software products: Commons Code C, FOP, Commons Beanutils Package, Commons Logging Package, Hivemind, Text ORO, Xalan-Java, Avalon, Batik, Catalina, Tomcat, Naming-Resources, log4j, and HTTP Client, (collectively, "Apache 2.0 Software").

Sterling Commerce has made no modifications to Apache 2.0 Software files.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes

of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally

submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent

to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

UTIL.CONCURRENT

The Sterling Commerce Software is distributed with or on the same storage media as the Util.Concurrent Software, located at $CMUHOME/javalib/concurrent1.3.2.jar which is subject to the following license:

All classes of util.concurrent release 1.3.4 were released to the public domain and may be used for any purpose whatsoever without permission or acknowledgment. [http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html].

Portions of the CopyOnWriteArrayList and ConcurrentReaderHashMap classes are adapted from Sun JDK source code. These are copyright of Sun Microsystems, Inc, and are used with their kind permission, as described in this license:

TECHNOLOGY LICENSE FROM SUN MICROSYSTEMS, INC.
TO DOUG LEA

Whereas Doug Lea desires to utilized certain Java Software technologies in the util.concurrent technology; and
Whereas Sun Microsystems, Inc. (“Sun”) desires that Doug Lea utilize certain Java Software technologies in the util.concurrent technology;

Therefore the parties agree as follows, effective May 31, 2002:
“Java Software technologies” means classes/java/util/ArrayList.java, and classes/java/util/HashMap.java.

Sun hereby grants Doug Lea a non-exclusive, worldwide, non-transferrable license to use, reproduce, create derivate works of, and distribute the Java Software and derivative works thereof in source and binary forms as part of a larger work, and to sublicense the right to use, reproduce and distribute the Java Software and Doug Lea's derivative works as the part of larger works through multiple tiers of sublicensees provided that the following conditions are met:

-Neither the name of or trademarks of Sun may be used to endorse or promote products including or derived from the Java Software technology without specific prior written permission; and

-Redistributions of source or binary code must contain the above copyright notice, this notice and the following disclaimers:
This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MICROSYSTEMS, INC. AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN MICROSYSTEMS, INC. OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN MICROSYSTEMS, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
You acknowledge that Software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

signed [Doug Lea]

dated

JAXB SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the JAXB Software, located at $CMUHOME/javalib/jaxb-api.jar, jaxb-libs.jar, jaxb-ri.jar, and jaxb-api.jar, which is subject to the following license:

COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0 (text)

* 1. Definitions.

o 1.1. Contributor means each individual or entity that creates or contributes to the creation of Modifications.

o 1.2. Contributor Version means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

o 1.3. Covered Software means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

o 1.4. Executable means the Covered Software in any form other than Source Code.

o 1.5. Initial Developer means the individual or entity that first makes Original Software available under this License.

o 1.6. Larger Work means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

o 1.7. License means this document.

o 1.8. Licensable means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

o 1.9. Modifications means the Source Code and Executable form of any of the following:

* A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

* B. Any new file that contains any part of the Original Software or previous Modification; or

* C. Any new file that is contributed or otherwise made available under the terms of this License.

o 1.10. Original Software means the Source Code and Executable form of computer software code that is originally released under this License.

o 1.11. Patent Claims means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

o 1.12. Source Code means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

o 1.13. You (or Your) means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, You includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, control means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

* 2. License Grants.
o 2.1. The Initial Developer Grant.
Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:
* (a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and
* (b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).
* (c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.
* (d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.
o 2.2. Contributor Grant.
Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:
* (a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and
* (b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).
* (c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.
* (d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.
* 3. Distribution Obligations.
o 3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

o 3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

o 3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

o 3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

o 3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipients rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

o 3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

* 4. Versions of the License.

o 4.1. New Versions.

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

o 4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

o 4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

* 5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN AS IS BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

* 6. TERMINATION.

o 6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

o 6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as Participant) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice

from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

o 6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

* 7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTYS NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

* 8. U.S. GOVERNMENT END USERS.

The Covered Software is a commercial item, as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of commercial computer software (as that term is defined at 48 C.F.R. 252.227-7014(a)(1)) and commercial computer software documentation as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

* 9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdictions conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree

that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.
* 10. RESPONSIBILITY FOR CLAIMS.
As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.
ZLIB SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the Zlib Software, located at $CMUHOME/<os>/bin/*ftp* and *ssh* which is subject to the following license:

zlib License
License
/* zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.3, July 18th, 2005

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu

LIBXML and LIBXSLT SOFTWARE:

The Sterling Commerce software is distributed with or on the same storage media as the LibXML software, and the LibXSLT Software, located at $CMUHOME/<os>/bin/cmusvwd, which are both subject to the following license:

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY WHETHER IN AN ACTION OF CONTRACT, TORT, OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

JDOM
The Sterling Commerce Software is distributed with or on the same storage media as the JDOM Software, located at $CMUHOME/javalib/jdom.jar, which is subject to the following license:

LICENSE

Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.

3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <request_AT_jdom_DOT_org>.

4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management <request_AT_jdom_DOT_org>.

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)." Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter_AT_jdom_DOT_org> and Brett McLaughlin <brett_AT_jdom_DOT_org>. For more information on the JDOM Project, please see <http://www.jdom.org/>.