Connect:Enterprise UNIX®

Integration Tools User's Guide

Version 2.4

Sterling Commerce
An IBM Company

Connect:Enterprise UNIX Integration Tools 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

Connect:Enterprise File Agent

Connect:Enterprise File Agent is the component of Connect:Enterprise that monitors specified directories for files, then begins processing those files. It provides consistent monitoring capability to Connect:Enterprise, which allows directories to become active transfer points for a specific database.

Connect:Enterprise File Agent Activity

The File Agent monitors user-defined staging directories for deposited files. The files are selected for processing according to the contents of the File Agent configuration file, a flat text file with filename patterns to watch for, and associated processing directives. The processing depends on whether the file is outbound (destined for a remote) or inbound (coming from a remote).

Outbound files are deposited into the staging directory by any application. After the File Agent detects the new file, it monitors the last-modified-time of the file and the file size to determine when the file is complete, then it performs the processing directives from the configuration file. The actual processing of the outbound file is done in the following sequence:

1. Add to the repository and optionally trigger an outbound connection.
2. Run an associated command, if specified.
3. Delete the file. The file is only deleted if the add to the repository and the command return a successful status.

Inbound batches are not actually added to the staging directory. Rather, the Batch Receive Informer (running in the Connect:Enterprise mailbox daemon) writes a zero-length file to the staging directory, whose file name contains the mailbox ID, batch number, batch ID, and other information about the batch. The File Agent determines the processing to do based on the information in the File Agent configuration file. The actual processing of the inbound file is done in the following sequence:

1. Run a command to extract the batch and do other back-end processing
2. Delete the zero length file. The file is only deleted if the command returns a successful status.

The File Agent gives you great flexibility and endless possibilities for automating data flow to and from your system. You define and modify these tasks through the File Agent configuration file.

Caution: Errors in the File Agent configuration file are fatal to the program. The checking parameter (–c) can be used to validate the configuration file.

The File Agent also has the capability of a direct API level interface with Connect:Enterprise for adding data files as batches to the Connect:Enterprise server.

If the API connection attempt to Connect:Enterprise fails, the connection is retried once per minute for 5 minutes, for a total of five retries. If all attempts fail, a critical error is posted to the log stream. The program immediately starts another 5-minute cycle. Normal processing is suspended until the connection to Connect:Enterprise is reestablished.

Logging is done to the File Agent *stderr* stream.

The file name matching mechanism is implemented with the UNIX regular expression functions. This gives you a great deal of flexibility when specifying file-matching criteria. The documentation for the regular expression syntax is available as part of the standard UNIX documentation in the regcomp(3c) and regex(5) manual pages.

Under normal conditions, each file is only processed once. The file is processed by the first configuration entry that matches the file name. You can use the regular expression-matching capabilities of the *file=* and *ignore=* entries to control the granularity of the selection, and ignore some files for selection entirely.

All files that begin with a period character, with the exception of a *.stop* file, are ignored by the File Agent. When the *.stop* file is detected, the program is terminated.

Existing files in the directory are re-recognized and reprocessed if their modification time or file size changes. This reprocessing allows you to accommodate remote sites repeatedly sending files of the same name.

Installing and Configuring Connect:Enterprise File Agent

Refer to the *Configuring Connect:Enterprise UNIX File Agent, MQ Agent, and cereport* section of the *Connect:Enterprise UNIX Installation and Administration Guide* for instructions on installing and configuring File Agent.

Designing and Planning Data Flows

Before you begin writing process commands in the configuration file, you must decide which directory and files you want the File Agent to monitor and the associated activities to perform when a file is identified for processing. You create a process line in the configuration file for each separate data flow.

Some of the questions you must answer include:

- Where is the data going?
- What is the process of the data transfer between the File Agent and the receiving application?

- When does the transfer occur?
- Do you want the File Agent to continue processing after an application or command is initiated or do you want the processing to pause until the initiated application or command is complete?
- How large are the data files? The File Agent monitors the directory periodically at specified intervals and for a number of cycles. What is a good time interval to check that the file is complete? Larger files require long periods of time and more cycles than shorter files.
- Do you want to delete the file after processing?
- Do you want to copy the file to other mailboxes?

Read the process lines and option lines in the following section. This gives you a clear picture of the types of information you must collect for the configuration file.

Before you execute the File Agent, you must create a File Agent configuration file, then check the configuration file syntax and program arguments.

Connect:Enterprise File Agent Configuration File

The File Agent configuration file is created during installation at $CMUHOME/etc/fileagt.cfg. It is a flat text file that can be modified with any text editor. You can modify the configuration file while the File Agent is running. The File Agent re-reads the updated configuration file without the need to stop and restart.

The File Agent configuration file can consist of comment lines, blank lines, option lines, and process lines. Option lines configure the File Agent. The process lines determine how files are processed. Physical lines can be continued in the usual UNIX nomenclature of a back slash character followed immediately by a new line character. The following sections list the options and requirements for comment lines, option lines, and process lines. Refer to *Sample Connect:Enterprise File Agent Configuration File* on page 12 for a sample File Agent configuration file.

Note: The MQWRITE functionality for MQSeries support is configured in the File Agent configuration file. Refer to the *Configuring MQWRITE* on page 28 for more information.

Editing the File Agent Configuration File

To make changes to the File Agent configuration file, use the following procedure to avoid termination of the File Agent:

1. Copy the configuration file to a temporary file.
2. Edit the temporary file.
3. Verify the changes using the following command:

```
ceufileagt -c -C temporary_file -l $CMUHOME/log/agent
```

where *temporary_file* is the file you created in step 1 on page 7.

4. Correct any errors that result from step 3 on page 7.
5. Repeat step 3 on page 7 and step 4 on page 8 until there are no errors.
6. Copy the contents of the temporary file into the active File Agent configuration file.

Comment Lines

Comment lines consist of a line beginning with a hash character (#). The remaining text up to the following new line character is ignored. Blank lines are ignored.

Option Lines

Option lines can consist of five different option fields, each of which can be overridden by the associated command line options to the program when it is started. If the associated command line option is specified, the option specified in the File Agent configuration file is ignored. For example, if –D *directory-path* is specified on the command line, any instance of *dir=directory-path* in the configuration file is ignored.

Option Line	Definition	Command Line Option
dir=directory-path	The full path of the directory where File Agent locates the files.	–D
cycles=nnn	Number of wait cycles to delay processing a file to ensure that the file is complete.	–W
interval=nnn	Number of seconds in a wait cycle.	–I
procs=nnn	Number of concurrent child processes the File Agent can have at one time.	–n
debug=nnn	Level of extended debugging information from the program. Use this option for debugging programs.	–d
cmu="host:port:id:pass"	The host name, TCP/IP port, user ID, and password needed to connect to the Connect:Enterprise server via TCP/IP. The user ID is used as the origination ID for all batches placed in the repository by the File Agent.	–M

Process Lines

Process lines consist of *file="expression"* or *ignore="expression"* fields followed by optional fields that specify the actions taken when the file is identified. When a file is identified, the File Agent can perform three tasks:

1. Add the file to the repository through the API. Turn off with id=none.
2. Run a command and wait for the program to complete, or initiate the command and continue running.

3. Delete the specified file and purge it from the file system.

All process line fields support one or a combination of these three tasks. The supporting process fields are displayed in the following table and defined in the following section.

Process Line Field	Data Direction	Function Performed
file="expression" or ignore="expression"	Both	Select a file by file name. Required line.
addcard="addcard string"	Outbound	Header record for API add. No Default.
bid="batch id string" or batchid="batch id string"	Outbound	Batch ID for batch corresponding to file. Default is name of matched file.
case=y|n	Both	Character case sensitivity on matching. Default is case=y.
cclist="mbox1,mbox2,mbox3,mbox4"	Outbound	List of carbon-copy mailboxes. Default is no cclist.
code=a|e|b	Outbound	Data type for added batches. Default is code=a.
command="shell command" or synccmd="shell command"	Both	Shell command specification.
delete=y|n	Both	Delete file after processing. Default is delete=n.
id=mailbox or mailbox=mailbox	Outbound	Mailbox ID to add file to. Turned off with *id=none.* Default is mailbox ID used for the Connect:Enterprise API.
id=none or mailbox=none	Inbound	Turns off the mailbox add step.
multxmit=y|n	Outbound	Batch MultiTransmit Flag. Default is multxmit=n.
trigger=y|n	Outbound	Auto connect trigger for contact = data_immediate in ACD. Default is trigger=n.

Process Line Field Descriptions

The following fields are allowed in process lines.

Field	Description
file="*expression*" or ignore="*expression*"	The *file* and *ignore* fields match file names in the staging directory. All process line definitions must start with either a *file*= field or an *ignore*= field. The File Agent processes any file matched with *file*= according to the rest of the process line. Any file matched by *ignore* is ignored and not processed. The *expression* is a regular-expression (RE) and must be enclosed in double quotes. The File Agent also has the capability of using RE substrings for substitution in the *id, bid, command,* and *synccmd* fields. The *ignore* field is useful when other programs have an internal naming convention for intermediate data files. For example, File Agent files are named *filename*.tmp when they are sent using FTP. When the transfer is successful, the file is renamed to *filename.* Specifying "ignore= "[.]tmp$" in the configuration file causes the File Agent to ignore the intermediate file names. **Note:** The specification *ignore="^[.]"* is a default in the program. Any file name beginning with a period character except **.stop** is ignored.
case = *y*|*n*	controls the case sensitivity when file names are matched. The default is *case=y*. If *case=n* is used, then file names are recognized without regard to upper and lowercase letters.
delete = *y*|*n*	allows File Agent to automatically purge the source file from the staging directory after processing if the associated value is set to *y*. The *delete* field cannot be used with a *command* field, but can be used in conjunction with a *synccmd* field.
addcard="*string*"	adds a data record at the beginning of a file as it is inserted into the repository as a batch. The program does not analyze the contents of the string, but binary zero values should be avoided. A typical use of *addcard* is to add a remote-specific record to the beginning of the data before transmission. In File Agent, this is a $$ADD control record. Normal ASCII characters are enclosed between the double quote delimiters ("") for the string value. Code special characters and binary values as shown in the following table: ◆ \r Specifies a carriage-return character - value is 0x0d ◆ \n Specifies a new line character - value is 0x0a ◆ \xdd Specifies a 2-digit hexadecimal character value ◆ \0dd Specifies a 2-digit octal character value **Note:** You must code the end-of-line terminator on the addcard. The program does not implicitly code one; the string is embedded as-is into the batch. **Caution:** The values coded in the string cannot include the value of binary zero for a character. This value is used as the string termination character and results in premature termination of the addcard string.

Field	Description
command="shell-command"	specifies a UNIX shell command. The command uses the UNIX system (3) command and processing continues immediately. That is, the child UNIX process runs asynchronously with the File Agent. The *command* field supports the following string substitution macros and is also eligible for substitution of RE subfields from the file field. ◆ %P—The %P token is replaced with the fully qualified file name of the staged file. ◆ %D—The %D token is replaced with the directory name of the staging directory. ◆ %F—The %F token is replaced with the simple file name of the staged file. ◆ %T—The %T token is replaced with the transaction number corresponding to the staged file. ◆ %L—The %L token is replaced with the directory name of the logging directory. For example, a command="/bin/cp %P /tmp/ftpdir/%F" with an input file name of "/opt/stercomm/stage/add.992341.08" results in a shell-command of "/bin/cp /opt/stercomm/stage/add.992341.08 /tmp/ftpdir/add.992341.08". **Caution:** Use of *delete=y* is not allowed with the *command* field because it would be possible for the File Agent to purge the staged file before the *shell-command* opened the file for processing.
synccmd="*shell command*"	specifies a UNIX shell command. The command uses the UNIX system (3) command, and the File Agent waits until the command is finished. That is, the child UNIX process runs synchronously with the File Agent. The *delete* field can be used with the *synccmd* parameter to purge the file after the command is finished. **Note:** If the program that the shell command is executing hangs when using the synccmd, the entire file agent daemon will hang until the hung program is resolved.
id=mailbox	specifies the destination mailbox ID for the file (outbound). The field defaults to the *remote ID* field of the –**M** command parameter or the coemption string. If *none* is specified as the destination mailbox (*id=none*), the file is not added to the repository. Any subsequent commands and deletions are processed normally. Use *id=none* to process files without adding them to the repository (inbound). The *id* field is eligible for substitution of RE subfields from the file field. However, only the first eight characters are used.
bid="mailbox batch id"	specifies the batch name to be used as the batch identifier when the file is inserted into the destination mailbox (id=*mailbox*). The value of the *bid* field defaults to the file name matched by the *file* field. The *bid* field is eligible for substitution of RE subfields from the file field.

Field	Description
multxmit=y|n	allows the multiple transmit (M) mailbox batch flag to be set on batches added to the repository. The default is multxmit=n.
cclist="mbox1,mbox2,mbox3"	designates a carbon copy list of mailbox ID fields that receive a copy of the batch being added to the mailbox designated in the ID field.
trigger=y|n	triggers a Connect:Enterprise auto connect session if File Agent is configured appropriately. The default is trigger=n.

Sample Connect:Enterprise File Agent Configuration File

A sample File Agent Configuration File is displayed on the following pages:

```
# ------------------------------------------------------------
# OPTION LINES
#
# Directory to watch for files to appear
dir      = /home/ceunix/fileagent/stage

# Connect:Enterprise UNIX signon info
cmu      = "server1:7557:qatest:qatest"

# Process files only after idle for 2 cycles of 15 seconds each
cycles   = 2
interval = 15

# Total number of concurrent inbound and outbound processes
procs    = 2

# ------------------------------------------------------------
# FILE RECOGNITION PATTERNS
#
# Watch the staging directory for files to appear that match
# any of the patterns below.  The first pattern to match is used.
#

# Ignore files ending in .tmp
# Note: if you allow FTP renames (the default in CEU 2.0 and above unless
# you code -K on your cmuftpd daemon startup), you will want to code
# the -e parm on your cmumboxd startup, to also kick off the Informer when
# the rename from filename.tmp to filename occurs.
#
ignore="[.]tmp$"
```

```
# INBOUND DATA FROM CEU VIA BATCH RECEIVE INFORMER
#
# Detect a touch file from the Batch Receive Informer of the format:
#
#    ce.batch_num.batch_file_size.mailbox_id.canonical_batch_id
#
# Example 1:
# Invoke the /home/ceunix/etc/eob-caller.sh script which then calls the
# /home/ceunix/eob.sh script with the fully qualified touch file name (%P)
# and the batch number (&1).  The eob-caller.sh script will delete
# the touch file after calling the eob.sh script.
#file="^ce[.]([0-9]+)[.]([0-9]+)[.](.*)[.](.*)" case=n id=none
command="/home/ceunix/etc/eob-caller.sh %P &1"

# Example 2:
# Invoke the /home/ceunix/etc/direct.sh script so Connect:Direct can
# process the inbound batch represented by the touch file.
# The direct.sh script is responsible for deleting the touch file.
#file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" case=n id=none
command="/home/ceunix/etc/direct.sh %P &1 %L %T"

# Example 3:
# Invoke the /home/ceunix/etc/mqwrite.sh script to send the inbound
# batch represented by the touch file to an MQ Series queue.
# The mqwrite.sh script is responsible for deleting the touch file.
# Note: synccmd allows only one process, regardless of procs= setting above.
#file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" id=none delete=y
synccmd="/home/ceunix/etc/mqwrite.sh -f %P -t %T -b &1 -l %L -q ENTERPRISE.LOCAL.QUEUE
-r ENTERPRISE.QUEUE.MANAGER -g yes -m text"
# OUTBOUND DATA FROM STAGING DIRECTORY TO CEU
#
# Detect a real file to be added to the Connect:Enterprise UNIX mailbox.
#
# Example 1:
# Add a file of the form "add_to_mailbox.anything" to the repository
#file="^add_to_mailbox.*" id=your_mailboxid_here code=b

# Example 2:
# Add a file from Connect:Direct to the repository
#file="(.*)[.]CCOD[.](.*)[.](.*)[.](.*)[.](.*)[.](.*)"  id=ceuser code=b
```

Checking Syntax and Program Arguments

After you create or modify the File Agent configuration file, check it for syntax errors and invalid program arguments. Perform this check each time you modify the File Agent configuration file.

To check the File Agent configuration file, type **ceufileagt –c –C fileagt.cfg -l $CMUHOME/log/agent** from the command line.

Errors are written to *stderr*. If no errors are found, the File Agent configuration file is ready to use in production.

Starting Connect:Enterprise File Agent

You can start the File Agent from the startup script or from the command line. The startup script is created during the installation procedure.

Starting Connect:Enterprise File Agent from the Startup Script

When the File Agent is installed, a startup shell script named *cefastartup.sh* is created in the $CMUHOME/etc directory. To run the script, type **cefastartup.sh** on the command line from the $CMUHOME/etc directory.

Starting Connect:Enterprise File Agent from the Command Line

Start the File Agent from the command line by using the **ceufileagt** command. The **ceufileagt** command must include the configuration file parameter (–C). You can add other parameters from the *Parameter Descriptions* on page 15.

To start the File Agent from the command line:

1. Start the File Agent server.
2. Type **ceufileagt –C fileagt.cfg** at the command line.

Using Connect:Enterprise File Agent Command Parameters

You can use command parameters to establish settings including a connection to the repository, waiting cycles, and directories. The following table contains a list of valid parameters for the **ceufileagt** command. For arguments that can be specified both from the command line and in the File Agent configuration file, the command line always takes precedence.

Parameter	Function	Can be specified in the File Agent configuration file?
–C *file name*	File Agent configuration file name	
–c	check configuration file	
–D *directory-path*	directory path	x
–d *level*	debugging level	x
–h	help	
–I *numbers*	number of wait cycles	x
–l *file name*	log file	

Parameter	Function	Can be specified in the File Agent configuration file?
–M *host name:ip-port:remote-id:password*	mailbox-designation	x
–n	sub-proc-limit	x
–W *seconds*	length of wait cycle	x

Parameter Descriptions

The following table describes each parameter:

Parameter	Description
–C *file name*	specifies the full path name to the File Agent configuration file for file name matching. This parameter can be specified only on the command line.
–c	checks the File Agent configuration file and program arguments, then exits the program. Use this parameter to validate File Agent configuration file changes before starting the program in production mode. Errors in parameters and the configuration file are logged. If errors exist, the program exits with a non-zero return code. If no errors exist, a zero return code is issued.
–D *directory-path*	specifies the full path to the file directory the File Agent monitors for files to arrive. This path should be a fully qualified path beginning with a back slash (/). You can specify this parameter in the File Agent configuration file with the *dir* field.
–d *level*	turns on logging of extended debugging information from the program. A high value for level results in more detailed diagnostics. This parameter is mostly for debugging the program during development. You can specify this parameter by using *debug=nnn* in the File Agent configuration file, but the command line takes precedence.
–h	prints a short-form help message concerning program arguments to the *stderr* of the program then exits the program.
–I *number*	specifies the number of full wait cycles the watch program delays processing a file to make sure that it is complete. This parameter can also be specified as *cycles* in the File Agent configuration file.
–l *file name*	specifies the full path for the file where the file agent log information is written. This parameter is required when checking the File Agent configuration file. **Note:** If an instance of File Agent is already running in a given directory, and you attempt to start an additional File Agent in the same directory, a log file with the name *filename.{pid}* is created and the contents of *filename.{pid}* will indicate which UNIX process is already running in the directory. The *{pid}* is the process ID assigned by Connect:Enterprise when attempting to create an additional instance File Agent.

Parameter	Description
–M *host name:ip-port:remote-id:password*	specifies the information for establishing the File Agent API session via TCP/IP to the Connect:Enterprise server. The specific elements are the host name of the server where the repository resides, the TCP/IP listening port used by Connect:Enterprise server on that host, the mailbox remote ID to use for the session, and the mailbox password to use for the session. The –**M** parameter must be specified with all 4 values. If an API session cannot be established, the program goes into a timer-retry session. It waits for 60 seconds and then tries to establish the session again. An error message is issued upon each failure. If the program fails five times in a row, the attempt is aborted, a critical error message is logged, and another wait *cycle* is observed before another connection attempt is made. If problems persist, verify that Connect:Enterprise is started. Other causes can be related to network or File Agent configuration. The File Agent attempts to contact the mailbox server at 1-minute intervals for 5 minutes before terminating an attempt. Another attempt is made at the next interval when the File Agent checks the directory.
–W *seconds*	specifies the number of seconds that constitute a wait cycle. The total time that a file must remain unmodified to be processed can be calculated as cycles times interval. For example, with –**I** 4 and –**W** 15, the file would have to be idle for a total of at least 60 seconds and a maximum of 75 seconds before processing begins. This parameter can also be specified as *interval* in the File Agent configuration file.

Stopping Connect:Enterprise File Agent

You can stop the File Agent by issuing the **fashutdown** command. The File Agent is normally stopped as part of **ceushutdown**.

fashutdown Format

During normal operations, the ceushutdown script calls the fashutdown script to shut down the File Agent. The default call to fashutdown uses the file $CMUHOME/etc/fileagent.cfg to locate the corresponding File Agent and terminate it.

Note: If you run more than one File Agent or if you use a configuration file other than $CMUHOME/etc/fileagent.cfg, you must modify the ceushutdown script.

You can request a shutdown of the File Agent using the following command:

```
fashutdown [immediate] [config-file|stage directory]
```

fashutdown Parameters

The **fashutdown** command has the following optional parameters:

Parameter	Description
immediate	with the immediate argument, the File Agent is terminated by signal (immediately). Otherwise, the File Agent is terminated gracefully. If the **-i** parameter is supplied to ceushutdown, then the immediate parameter is used with fashutdown.
config-file or stage directory	with this argument, the fashutdown script terminates the corresponding instance of File Agent.

Configuring Connect:Direct to Connect:Enterprise File Agent Notification

You can configure Connect:Direct to notify the File Agent whether a transferred file is complete or incomplete. If a file is incomplete, the File Agent does not add it to the Connect:Enterprise repository. To do this, you must change the Connect:Direct Initialization Parameter file and the File Agent configuration file.

Updating the Connect:Direct Initialization Parameter File

Use the following steps to update the Connect:Direct Initialization Parameter file:

1. Open your Connect:Direct **initparm.cfg** file.
2. Set the **:file.agent field** to **y**. This setting will ensure that the file is renamed after it is added to the staging directory according the pattern specified in *Updating the Connect:Enterprise File Agent Configuration File* on page 17. If you set this parameter to **n**, the file is still added, but is not renamed.
3. Add the File Agent staging directory to **:staging.directory**. An example follows:

```
# File Agent For Connect:Enterprise UNIX
file_agent:\
  :file.agent=y:\
  :staging.directory=/domain/users/user01/CEnterprise/ceu/stage:
```

The value for the staging.directory parameter is the full path to the File Agent staging directory.

Updating the Connect:Enterprise File Agent Configuration File

In the File Agent configuration file, set the **file** parameter to match a pattern as follows:

```
file="<filename>.CCOD.<feedback code>.<condition code>.
<Connect:Direct node name>.<process record or event record>
```

Note: The regular expression of this parameter is: file="(.*)[.]CCOD[.](.*)[.](.*)[.](.*)[.](.*)[.](.*)"

Following is an example:

```
file="cddelete.me.CCOD.0.0.cdunix.node.P"
```

Updating the Connect:Direct Process

After you update the File Agent configuration file, you need to include the path to the File Agent staging directory in your Connect:Direct Process. Following is an example:

```
sample process snode=snode_cdu3600

step01
  copy from (file = /domain/users/user01/CDirect/cdu3600/ndm/bin/direct
             pnode
            )
       ckpt = 2M
       compress = extended
       to   (file = /domain/users/user01/CEnterprise/ceu/stage/cddelete.me
             snode
             disp = new
            )

pend;
```

Startup

In order for the Connect:Direct to File Agent notification to work, you must start the components in the following order:

1. Connect:Enterprise server
2. Connect:Enterprise File Agent
3. Connect:Direct server

Managing File Agent Log Files

You can manage File Agent log files by moving, renaming, or deleting the File Agent log using the UNIX file system. When you move, rename, or delete a log file, a new file is automatically created.

The File Agent log files are located in the following directory:

```
$CMUHOME/log/agent
```

Chapter 2

Understanding the Batch Receive Informer

The Batch Receive Informer is part of the mailbox daemon and is notified whenever a batch is added to the repository. Use the Batch Receive Informer to designate which batches are to be processed by the File Agent and to notify the File Agent that a batch matching the preset criteria has been received.

This chapter discusses the following topics:

- Understanding the Batch Receive Informer functionality
- Setting the CE_BATCHRECEIVE variable
- Creating and Modifying the Batch Receive Informer configuration file

Batch Receive Informer Functionality

The Batch Receive Informer, a component of Connect:Enterprise, handles the interaction between Connect:Enterprise and the File Agent. The Batch Receive Informer uses a configuration file to designate the criteria by which batches are selected for the File Agent to process. After a batch is added to the repository that matches the criteria, the Batch Receive Informer writes a zero-length file to the staging directory monitored by the File Agent.

The name of the zero-length file written by the Batch Receive Informer is in the format **ce.*batch_number*.*batch_file_size*.*mailbox_id*.*canonical_batch_id***, where:

Section	Description
ce	is the prefix for all zero-length files created by the Batch Receive Informer.
batch_number	represents the batch number.
batch_file_size	represents the size of the batch.
mailbox_id	represents the mailbox ID assigned to the batch.
canonical_batch_id	represents the batch ID with all special characters replaced with the underscore (_) character.

Setting the CE_BATCHRECEIVE Variable

The CE_BATCHRECEIVE environment variable specifies the location and name of the Batch Receive Informer configuration file. CE_BATCHRECEIVE must be added to the **ceustartup** script, the script used to start Connect:Enterprise, to define a location other than the default location.

Perform the following procedure to add CE_BATCHRECEIVE to your **ceustartup** script:

1. Open **ceustartup**.

```
vi ceunix/etc/ceustartup
```

2. Set the CE_BATCHRECEIVE variable.

```
CE_BATCHRECEIVE="file name"
```

The Batch Receive Informer searches for the file in the $CMUHOME/etc directory unless an absolute path name is defined. If CE_BATCHRECEIVE is not set, the Batch Receive Informer looks for $CMUHOME/etc/ce_receive.cfg.

For example:

```
#!/bin/ksh
#
# This script will bring up Connect:Enterprise UNIX.
# Each daemon will be started on the host that was specified
# during the Connect:Enterprise UNIX customization procedure.
#
# Please note that this sample script should be reviewed and
# edited as necessary before starting Connect:Enterprise UNIX.
# As a default, this sample, as created at the time of installation,
# assumes that all the daemons will be running locally, not distributed.
# The commented lines, below, illustrate these same daemons being
# started on remote hosts:
#
#
export CE_BATCHRECEIVE="informer.cfg"
cd /home/fremont6/user1/enterprise/ceunix/trace
"/home/fremont6/user1/enterprise/ceunix/sun/bin/cmuctld" -P 17500 &
sleep 1
"/home/fremont6/user1/enterprise/ceunix/sun/bin/cmumboxd" -H georges -P 17500 &
sleep 1
"/home/fremont6/user1/enterprise/ceunix/sun/bin/cmuacd" -H georges -P 17500 &
sleep 1
"/home/fremont6/user1/enterprise/ceunix/sun/bin/cmulogd" -H georges -P 17500 &
sleep 1
"/home/fremont6/user1/enterprise/ceunix/sun/bin/cmuexitd" -H georges -P 17500 &
sleep 1
# "/home/fremont6/user1/enterprise/ceunix/sun/bin/cmubscda" -H georges -P 17500 &
sleep 1
# "/home/fremont6/user1/enterprise/ceunix/sun/bin/cmuasyd" -H georges -P 17500 &
sleep 1
"/home/fremont6/user1/enterprise/ceunix/sun/bin/cmuftpd" -H georges -P 17500 &
sleep 1
#
# End of the Connect:Enterprise UNIX startup script.
#
```

Creating and Modifying the Batch Receive Informer Configuration File

The Batch Receive Informer configuration file is a flat text file that can be modified with any text editor. The Batch Receive Informer configuration file can be updated without stopping and restarting Connect:Enterprise.

Each line of the Batch Receive Informer configuration file consists of four parts—mailbox ID, protocol, batch ID, and staging directory. In order for a batch to be processed by the File Agent, the mailbox ID, protocol, and batch ID associated with the batch must match those designated in the Batch Receive Informer configuration file. When a batch is received that meets all three criteria, a notification is placed in the staging directory. The File Agent monitors the staging directory and processes the appropriate batches as the notifications appear.

The format of each line of the Batch Receive Informer configuration file is as follows:

```
mailboxID protocol batchID staging_directory
```

The following table describes each element:

Element	Description
mailboxID	represents the 1–8 character mailbox ID associated with the batches you want processed by the File Agent. Shell File Name Matching patterns are allowed, for example, *,?,[ ].
protocol	represents the communications protocol associated with the batches you want processed by the File Agent. Valid values for protocol are **async**, **bisync**, **ftp**, **offline**, **ssh**, **gis**, **http**, **ediint** or the asterisk (*****) wildcard character indicating all protocols. The bang (**!**) wildcard character followed by a protocol value is also valid to indicate all protocols excluding the protocol specified. For AS2, use **http** to match inbound AS2 messages and MDNs. Use **ediint** to match outbound AS2 messages, MDNs, and inbound AS2 payloads.
batchID	represents the 1–8 character batch ID associated with the batches you want processed by the File Agent. Shell File Name Matching patterns are allowed, for example, *,?,[ ]. The value must be enclosed in quotation marks. **Caution:** Using shell-sensitive metacharacters in the batch ID string can cause problems.
staging_directory	represents the absolute path name of the directory that the File Agent is watching for notifications that a batch needs to be processed by the File Agent. The notifications appear as zero-length files where the file name is the metadata transference method**.** **Note:** The value of this parameter must be the same as the value of the **dir** parameter in the File Agent configuration file.

The Batch Receive Informer configuration file can consist of as many definition lines as needed. The first line to match an incoming batch is used. Comment lines begin with # and blank lines are allowed.

Sample Batch Receive Informer Configuration File

```
#  Filter lines below; First one that matches is used.
#
# The first two lines direct all AS2 inbound payload and MDN
# notifications to a special File Agent stage directory.
#  All other AS2 batches and those from cmuadd are ignored.
# Notifications for the remaining batches are sent to the
# normal  File Agent stage directory.
# This allows two different instances of the File Agent
# to process the work separately.
#
# ID    Protocol         Batch ID    Stage directory to put "touch file" in.

*       ediint           '*.OK'      /home/ceunix/fileagent/stage_AS2_inbound
*       http             '*.MD'      /home/ceunix/fileagent/stage_AS2_inbound

*       !offline,!ediint,!http '*'   /home/ceunix/fileagent/stage
```

Chapter 3

MQSeries Support

Connect:Enterprise provides basic integration with IBM MQSeries. Users can retrieve data from MQSeries queues and write it to the Connect:Enterprise repository. This functionality is referred to as MQREAD and involves the Message Agent (Magent) component. With the MQWRITE functionality, users can retrieve data from the repository and write it to an MQSeries queue. MQWRITE is implemented as part of the existing Connect:Enterprise File Agent component.

System Requirements

IBM MQSeries version 5.1 or later is required for Connect:Enterprise MQSeries support. Only MQSeries Type 2 headers are supported. Connect:Enterprise MQSeries support is available on all AIX, HP-UX, and Solaris platforms as described in the *Connect:Enterprise UNIX Version 2.3.00 Release Notes* with the exception of Solaris 8. MQSeries is not certified for the Solaris 8 operating system.

Configuring Connect:Enterprise File Agent

Refer to the *Configuring Connect:Enterprise UNIX File Agent, MQ Agent, and cereport* section of the *Connect:Enterprise UNIX Installation and Administration Guide* for instructions on configuring MQ Agent.

Message Agent (Magent)

Magent monitors for data to arrive on a configured MQSeries message queue. When a message is present, MQREAD reads the message (a single message, multiple message segments, or grouped messages) and writes the message data to a new batch in the Connect:Enterprise repository.

Configuring the Message Agent

The Message Agent configuration file is created during installation at $CMUHOME/etc/mqagent.cfg. It is a flat text file that can be modified with any text editor. You can modify the configuration file while the Message Agent is running. The Message Agent re-reads the updated configuration file without the need to stop and restart.

The Message Agent configuration file can consist of comment lines, option lines, global lines, process lines, and blank lines. Physical lines can be continued in the usual UNIX nomenclature of a back slash character followed immediately by a new line character. The following sections list the options and requirements for comment lines, option lines, global lines and process lines.

Comment Lines

Comment lines consist of a line beginning with a hash (#) character. The rest of the text up to the following new line character is ignored. Blank lines are ignored.

Option Lines

Option lines define access to the repository. Option lines can consist of one option field that can be overridden by the associated command line option to the program when it is started.

Option Line	Definition	Command Line Option
cmu="host:port:id:pass"	The host name, TCP/IP port, user ID, and password needed to connect to the Connect:Enterprise server via TCP/IP. The user ID is used as the origination ID for all batches placed in the repository by the File Agent.	–M

Global Lines

Global lines apply to all process line entries. These options are not available from the command line.

Parameter	Value(s)	Description
qmname	queue manager name	Specifies the queue manager name that the Message Agent accesses when processing messages on the defined message queues. If not supplied, the Message Agent uses the default queue manager defined on the system.

Process Lines

The process lines define each queue to be monitored. Process lines consist of **mqname=** followed by optional fields that specify the actions taken for the defined message queue.

The following table lists each parameter and its description that is passed from Magent to MQREAD. Required parameters are in bold. Default values are underlined.

Parameter	Value(s)	Description
mbid	**mailbox ID**	Specifies the mailbox ID where the batches should be created.
mqname	*queue name*	Name of MQSeries queue to monitor.
moption	group|count	Specifies if and how multiple messages are concatenated into a single batch. The value **group** indicates that batches are created according to the MQSeries message groups. Messages belonging to the same message group are concatenated into a single repository batch. Messages that arrive ungrouped are written to separate batches containing only the data from that message. The value **count** indicates that messages are concatenated into a single batch based on the **mcount** value. MQREAD retrieves **mcount** messages one at a time from the queue and concatenates their data into a single batch. If the queue is empty before **mcount** messages are retrieved, then the batch is created with data from fewer than **mcount** messages. The queue is considered empty if no message arrives for **mdelay** seconds.
mode	text|stream	Specifies whether or not to append new line characters to the message data. The value **text** indicates that a new line character is appended to the data of each message received before it is written to the repository. In this mode, the repository batch is written with a new line character following the data of each message. The **pad** and **padlen** parameters are ignored.
mcount	number of messages	Specifies the maximum number of messages to concatenate into a single batch. The default value is 10000. The parameter **mcount** is only used when **moption=count**.
mdelay	number of seconds	Specifies the number of seconds to delay before assuming that the queue is empty. The parameter **mdelay** is only used when **moption=count**. The default value of **0** indicates that MQREAD should close the batch immediately if the queue is emptied before **mcount** messages have been retrieved.
pad	character	Specifies the character used to pad message data before it is written to the repository. Padding is not performed if **pad** is not specified. This parameter is ignored when **mode=text**.
padlen	length	Specifies the length to pad message data before it is written to the repository. This parameter is ignored when **mode=text**.
qmname	queue manager name	Specifies the queue manager name that the Message Agent accesses when processing messages on the defined message queues. If not supplied, the Message Agent uses the default queue manager defined on the system.
bid	batch ID	Specifies the batch ID for the created batch.
UID	user ID	Specifies the user ID for the Connect:Enterprise connection.
upwd	user password	Specifies the user password for the Connect:Enterprise connection.

Configuration Example

The following is a sample **mqagent.cfg** file:

```
#
# The "cmu=" option line specifies the host, port, user ID, and password to be used
# when connecting to the mailbox to add batches.  If the password is
# not in the config file, then it will be prompted for at startup time.
#
cmu="localhost:8100:mailboxuid:mailboxpwd"
#
# The "qmname=" is an optional entry.  If not supplied, the Message Agent will use the
# default queue manager defined on the system.  It defines the queue manager name
# Message Agent will access when processing messages on the defined message queues.
#
qmname=enterprise.queue.manager
#
# Monitor an MQSeries queue called 'Queue01' on a queue manager defined by the qmname
# entry.  When data arrives create batches in the repository with mailbox ID
# 'Mailbox01' and user batch ID 'Queue01'.  Use the MQSeries message
# grouping to determine which messages go together into a particular batch.
#
mqname=Queue01 mbid=Mailbox01 bid=Queue01
#
# Monitor an MQSeries queue called 'Queue02' on the Queue manager defined by the
# qmname entry above.  When data arrives create batches in the repository with mailbox
# ID 'Mailbox02' and pad the message data to a length of 8000 with ASCII spaces, and
# create a new batch for each message.
#
mqname=Queue02 mbid=Mailbox02 pad=20 padlen=8000 moption=count mcount=1
```

Starting and Stopping the Message Agent

Magent is started and stopped using scripts. After the agents are configured using **ceuagtcust**, these scripts are invoked as part of the Connect:Enterprise **ceustartup** and **ceushutdown** scripts. Magent can also be started and stopped from the command line.

Configuring the Message Agent Startup Script

Before you run the startup script, configure the Magent parameters in the script. The following parameters are available:

Parameter	Value	Description
p	none	Indicates that the Connect:Enterprise password will be read from standard input.
d	/*pathname*	Turns on debugging. *pathname* is optional. You can specify a fully qualified pathname to specify the trace file. If you do not provide a pathname, the default of $CMUHOME/etc/ma.trace is used.
h	none	Print help and exit.

Parameter	Value	Description
M	“host:port:login[:password]”	Specifies the Connect:Enterprise connection information. If you include this information, the value of the “cmu=” configuration variable in the configuration file is ignored.
C	*path to mqagent.cfg*	Specifies the path of the mqagent.cfg file. If you do not provide this path, the default of $CMUHOME/etc/mqagent.cfg is used.

Following is an example startup script. Items in bold identify where the parameters are added:

```
#!/bin/ksh
#
# MQ Agent Startup Script - Installed InstallationDateTime
#
mqagent=/users/l2/ce20/aix/bin/magent

if [ "X$1" != "X" ]; then
echo $1 | $mqagent -d -p
else
$mqagent -d
fi
```

Starting the Message Agent Using the Startup Script

The Message Agent is normally started as part of **ceustartup** using the **cemqstartup.sh** script. The system administrator password is required at startup. The password can be specified in the **mqagent.cfg** file or the startup script prompts for the password.

Starting the Message Agent from the Command Line

To start the Message Agent from the command line, enter the following command:

```
$ cemqstartup.sh
```

There are no parameters associated with **cemqstartup.sh**.

Stopping the Message Agent Using the Shutdown Script

The Message Agent is normally shut down as part of **ceushutdown**. The Message Agent shutdown script sends a SIGTERM to the Message Agent so that all currently running processes are completed prior to shutdown.

Stopping the Message Agent from the Command Line

To stop the Message Agent from the command line, enter the following command:

```
$mashutdown
```

There are no parameters associated with the **mashutdown** script.

Message Agent Logging

The Message Agent (MQREAD) writes event messages to the File Agent log file at $CMUHOME/log/agent/cfa.log. A single record is written for each repository batch created by MQREAD. Each MQREAD record has a unique identifier for common reporting.

MQWRITE

MQWRITE reads a stream file and writes messages to a configured MQSeries message queue. The **mqwrite.sh** script extracts data from theConnect:Enterprise repository based on a file name created by the Batch Receive Informer. After extracting the data to a temporary file, the script calls the MQWRITE executable. MQWRITE reads the file, then writes the file as messages according to the parameters defined in the File Agent configuration file.

The functionality of MQWRITE accommodates two general data formats in stream files. The first format is one continuous stream of bytes as might be represented by an executable binary file. The second format is a stream file containing one or more logical records identified by a record delimiter or byte count.

Configuring MQWRITE

The MQWRITE feature is configured in the File Agent configuration file, which is located at $CMUHOME/etc/fileagt.cfg.

The following table lists each MQWRITE configuration parameter and its description. Required parameters are in bold. Default values are underlined.

Parameter	Value(s)	Description
–f	**file name**	Specifies the full path and file name. It is specified as –f %P in the UNIX shell command in the configuration file.
-F	n | s	n–specifies that the data format for the MQ Series message will be set to MQMD_FMT_NONE. This is the default. s–specifies that the data format for the MQ Series message will be set to MQMD_FMT_STRING.
–t	**transaction number**	Specifies the transaction number. It is specified as –t %T in the UNIX shell command in the configuration file.
–b	batch number	Specifies the batch number. It is specified as –b &1 in the UNIX shell command in the configuration file.
–l	**logfile directory name**	Specifies the MQWRITE log file directory name. It is specified as –l %L in the UNIX shell command in the configuration file.
–q	**queue name**	Specifies the message queue name where messages are written.

Parameter	Value(s)	Description
–r	*queue manager name*	Specifies the local MQSeries queue manager to which MQWRITE connects. The default value is the default local queue manager configured with MQSeries.
–s	max size	Specifies the maximum data size of the messages. When –q specifies a local queue, a value of 0 indicates that the MaxMsgSize defined for the queue is used. The 0 value is invalid when –q specifies a local definition of a remote queue. The default value is 10000. This parameter is ignored when –m text is specified and –q specifies a local queue.
–g	y|n or yes|no	Specifies if messages from a file are written as a message group. Messages assigned to a message group can be retrieved from the final destination queue as a group in the original sequence. The default value is yes.
–d	*file name*	Indicates to write debugging information to the given file name.
–m	s|t or stream|text	Indicates if the file information is read as a continuous stream or one line at a time based on new line characters. See the *Continuous Stream File* on page 30 and *Logical Record Stream File* on page 30 for more information.
-h or -?		Displays usage information.

MQWRITE Configuration Example

The following sample excerpts from **fileagt.cfg** show the MQWRITE information:

```
#
# Filenames of the format created by the Batch Receive Informer are matched
# It is not added to the repository
# The file is passed to MQSeries mqwrite.sh to be processed
# The file is deleted after processing by mqwrite.sh
#
# -f %P  The %P token is replaced with the fully qualified file name of the staged
# file
# -t %T  The %T token is replaced with the transaction number corresponding to the
#        staged file
# -b %1  The %1 token is replaced with the first field in the regular expression
#        parsed in the -f statement.  In this example it is (.*)
# -l %L  The %L token is replaced with the directory path of the MQWRITE log file
# -q ENTERPRISE.LOCAL.QUEUE  Specifies the queue name where the staged file will be
#                            written
# -r ENTERPRISE.QUEUE.MANAGER  Specifies the name of the queue manager on the local
#                              system where MQWRITE will connect
# -g yes  Specifies that the messages written to the queue are to be grouped
# -m text  Specifies that the file data is to be interpreted as text with native
#          new line controls
#
file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" id=none delete=y \
     synccmd="mqwrite.sh -f %P -t %T -b &1 -l %L \
     -q ENTERPRISE.LOCAL.QUEUE -r ENTERPRISE.QUEUE.MANAGER -g yes -m text"
```

```
#
# Filenames of the format created by the Batch Receive Informer are matched
# The file is passed to MQSeries mqwrite.sh to be processed
# The file is deleted after processing by mqwrite.sh
#
# -f %P  The %P token is replaced with the fully qualified file name of the staged
# file
# -t %T  The %T token is replaced with the transaction number corresponding to the
#        staged file
# -b %1  The %1 token is replaced with the first field in the regular expression
#        parsed in the -f statement.  In this example it is (two[.]three)
# -l %L  The %L token is replaced with the directory path of the MQWRITE log file
# -q ENTERPRISE.LOCAL.QUEUE  Specifies the queue name where the staged file will be
#                            written
# -r ENTERPRISE.QUEUE.MANAGER  Specifies the name of the queue manager on the local
#                              system where MQWRITE will connect
# -s 8000  Specifies 8000 bytes as the maximum message size
# -g yes  Specifies that the messages written to the queue are to be grouped
# -m stream  Specifies that the file data is to be interpreted as a continuous stream
#
file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" id=none delete=y \
     synccmd="mqwrite.sh -f %P -t %T -b &1 -l %L -q ENTERPRISE.LOCAL.QUEUE \
     -r ENTERPRISE.QUEUE.MANAGER -s 8000 -g yes -m stream"
```

Continuous Stream File

A continuous stream file is written as one message to the message queue up to the specified maximum message size. When the file exceeds the maximum message size of the queue, the file is divided into multiple messages.

To write messages from a continuous stream file, set **–m stream** and **–s *maxsize***, as in the following example:

```
file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" id=mailbox01 delete=y \
     synccmd="mqwrite.sh -f %P -t %T -b &1 -l %L -q ENTERPRISE.LOCAL.QUEUE \
     -r ENTERPRISE.QUEUE.MANAGER -s 8000 -g yes -m stream"
```

Logical Record Stream File

Logical records in a stream file can be one of three different types:

- Same-length records

  The records within a file, though not necessarily in the same format, are all the same length.

  To write messages from a file containing same-length records, set **–m stream** and **–s *record size***, as in the following example:

```
file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" id=mailbox01 delete=y \
     synccmd="mqwrite.sh -f %P -t %T -b &1 -l %L -q ENTERPRISE.LOCAL.QUEUE \
     -r ENTERPRISE.QUEUE.MANAGER -s 5000 -g yes -m stream"
```

- Variable-length records

  The records within a file are not the same length and are not delimited. You can read the variable-length record file into messages using the stream mode to create fixed-length messages of a size specified by the **–s** parameter. Variable-length records do not translate smoothly into messages. It can be difficult to reconstruct the original records from the message data because individual records are split between messages.

- Delimited records

  The records within a file end in a new line character. MQWRITE can be configured to read a line of data and write that line as a separate message to the message queue. The new line character is stripped from the record and is not written to the message. If a line exceeds the maximum message size defined for the queue, then the record data is separated into additional messages as needed and a warning is written to **mq.log**.

  To write messages from a file containing delimited records, set **–m text** as in the following example:

```
file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" id=mailbox01 delete=y \
     synccmd="mqwrite.sh -f %P -t %T -b &1 -l %L -q ENTERPRISE.LOCAL.QUEUE \
     -r ENTERPRISE.QUEUE.MANAGER -g yes -m text"
```

Note: The **–s** parameter is optional for writing messages from delimited records.

MQWRITE Logging

MQWRITE log information is written to $CMUHOME/log/agent/mq.log. This file contains the results of writing messages to message queues. Common reporting uses a unique transaction number to match entries in **cfa.log** with entries in **mq.log**.

Chapter 4

Connect:Enterprise Common Reporting

Connect:Enterprise uses scripts to link Connect:Enterprise, Connect:Direct, and File Agent components. Connect:Enterprise common reporting allows you to track data movement within all three components based on a file name, transaction number, or time stamp.

This chapter contains a description of how the scripts work together and instructions on configuring, installing, and performing common reporting.

File Navigation in Connect:Enterprise

The direction of the data flow determines how files navigate through Connect:Enterprise.

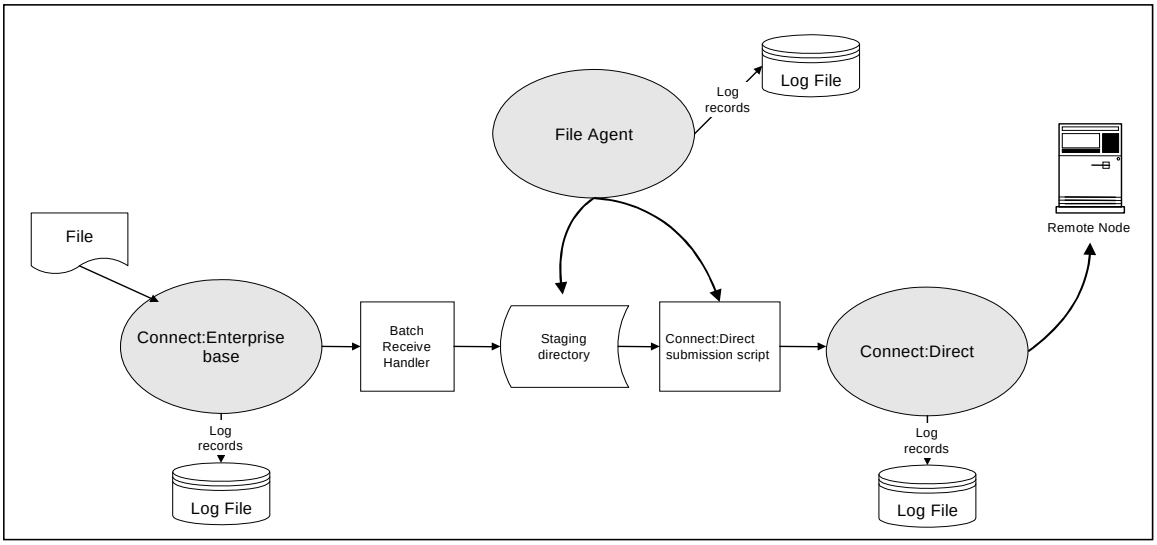
Connect:Enterprise to Connect:Direct

When a file targeted for Connect:Direct is added to Connect:Enterprise, the Batch Receive Informer creates a zero length notification file named ce.*batch_number.batch_file_size.mailbox_id.batch_id*, in a staging directory monitored by Connect:Enterprise File Agent.

When Connect:Enterprise File Agent detects the notification file in the staging directory, it calls a script, **direct.sh**, that sends the file to Connect:Direct.

At each step in this process, the system writes to log records that can be viewed with the **cereport** command.

The following illustration shows file navigation from Connect:Enterprise base to Connect:Direct.

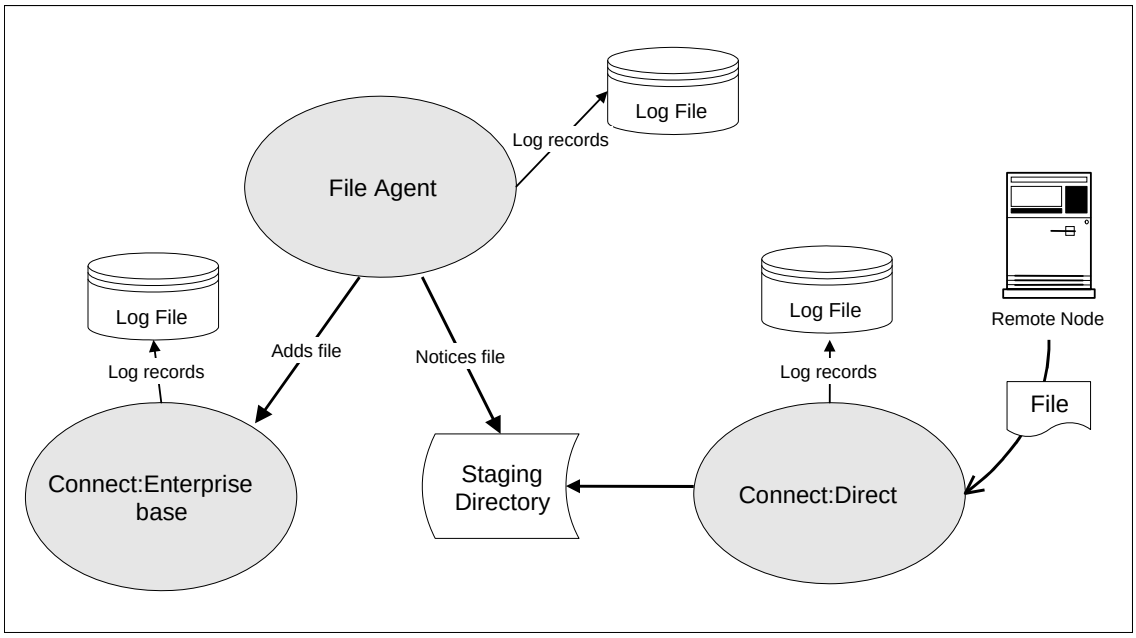


Connect:Direct to Connect:Enterprise

When Connect:Direct receives a file from the remote node, it copies the file into the staging directory monitored by the File Agent. The File Agent adds the file to Connect:Enterprise. Records created for each step of the process can be viewed using the **cereport** command.

A file moving from Connect:Direct to Connect:Enterprise base is handled differently. When Connect:Direct successfully copies a file, it sends a copy completion log message to the statistics file. This alerts File Agent, which then notifies Connect:Enterprise to extract the file from Connect:Direct. Log records are created at each step in this process. View the logging information with the **cereport** command.

The following illustration shows file navigation from Connect:Direct to Connect:Enterprise.



Configuring Connect:Enterprise File Agent

Before using the Connect:Enterprise File Agent, perform the following procedures to modify the File Agent configuration files to allow common reporting.

Configuring File Agent from the Installation Script

Refer to the *Configuring Connect:Enterprise UNIX File Agent, MQ Agent, and cereport* section of the *Connect:Enterprise UNIX Installation and Administration Guide* for instructions on configuring cereport.

Modifying the File Agent Configuration File

The File Agent configuration file tells Connect:Enterprise File Agent which action should be taken on a particular file type. Connect:Enterprise File Agent reads a file recognition pattern and executes the command immediately following the pattern. Following is a sample configuration file. Use a text editor such as vi to open and edit the **fileagt.cfg file**. The file is located in $CMUHOME/etc/fileagt.cfg. Before editing this file, refer to *Editing the File Agent Configuration File* on page 7.

```
# Installed Wed Oct 13 11:00:26 CDT 2004
#
# Directory to watch
dir      = /home/users/enterprise/ceunix/fileagent/stage
cmu      = "ceusun:17500:freds:freds"
cycles   = 2
interval = 15
procs    = 2
#
# file recognition patterns go below
#
# Send a file from Connect:Enterprise to Connect:Direct
# file="^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)" case=n id=none \
        command="/$CMUHOME/etc/direct.sh %P &1 %L %T"

# Add a file to the repository without cereport visibility
# file="^add_to_mailbox.*" id=mailboxID code=b

# Add a file from Connect:Direct to the repository with cereport visibility
# file="(.*)[.]CCOD[.](.*)[.](.*)[.](.*)[.](.*)[.](.*)" id=ceuser code=b

# ignore files ending in .tmp \(example\)
ignore="[.]tmp$"
```

This file contains three file recognition patterns with associated commands. They are:

Pattern	Command	Description
^ce[.]([0-9]+)[.]([0-9]+)[.]([^.]+)[.](.*)	*/$CMUHOME*/etc/direct.sh %P &1 %L %T	When one or more files match the pattern, File Agent executes the */$CMUHOME*/etc/direct.sh %P &1 %L %T command, sending matching files from the repository to Connect:Direct.
^add_to_mailbox.*	id=*mailboxID* code=b	When one or more files match the pattern, File Agent adds the matching files to the mailbox specified by id= . If you use this pattern, the transfer does not appear when the **cereport** command is issued.
(.*)[.]CCOD[.](.*)[.](.*)[.](.*)[.](.*)[.](.*)	id=*mailboxID* code=b	When one or more files match the pattern, File Agent adds the matching files to the mailbox specified by id= . If you use this pattern, the transfer does appear when the **cereport** command is issued.

Perform one or more of the following:

- To send a file from Connect:Enterprise using Connect:Direct, uncomment the first file recognition pattern and specify the location of you $*CMUHOME* directory.
- To add a file to the repository that is not visible to cereport, uncomment the second file recognition pattern and specify the mailbox ID to place the file (id=).
- To add a file to the repository from Connect:Direct that is visible to cereport, uncomment the third file recognition pattern and specify the mailbox ID to place the file (id=).

Create and Modify the Connect:Direct Submission Script

The Connect:Direct Submission script sends Connect:Enterprise files to Connect:Direct. Create and modify this file to specify a destination node for the transferred files.

Use the following procedure:

1. Copy the **sample_direct.sh** file as **direct.sh** into the ceunix/etc directory.

```
cd ceunix/etc
cp sample_direct.sh direct.sh
```

2. Use a text editor such as vi to open the **direct.sh** file.

```
vi direct.sh
```

3. Change the snode line to point to the target snode. In the following example, cd_node is the target snode.

```
snode="cd_node"
```

4. Change snodeid to a Connect:Direct remote user ID. In the following example, cd_user is the snodeid.

```
snodeid="cd_user"
```

5. Change snodepass to a Connect:Direct remote user password. In the following example, direct_password is the snode password.

```
snodepass="cd_password"
```

Note: If no password is required for the snodeid specified in step 4, include only the quotation marks. For example, **snodepass=""** .

6. Change snodedir to a directory on a remote server where files are placed after copy. In the following example, /home/cdunix/incoming/ is the snode directory:

```
snodedir="/home/cdunix/incoming/"
```

7. Change directbase to point to a local copy of Connect:Direct. In the following example, /home/cdunix/ndm is the directory location of a local copy of Connect:Direct.

```
directbase="/home/cdunix/ndm"
```

The following example illustrates the changes made in the preceding steps:

Note: This sample script is written to send a file from a UNIX operating system to a UNIX operating system. When sending to or from operating systems other than UNIX, you may need to change the **to** and **from** portion of the **copy** statement to compatible syntax.

```
#!/bin/ksh
#================================================================
#
# Connect:Enterprise UNIX (tm)
# COPYRIGHT (C) 1999
# Sterling Commerce, Inc.
# All Rights Reserved
#
# This material is the confidential trade secret and proprietary
# information of Sterling Commerce, Inc. and/or its subsidiaries.
# It may not be reproduced, used, sold, or transferred to any
# third party without the prior written consent of Sterling
# Software, Inc.  All rights reserved.
#
# Use, duplication, or disclosure by the Government is subject
# to restrictions as set forth in subparagraph (c)(1)(ii) of
# the Rights in Technical Data and Computer Software clause at
# FAR 52.227-7013.
#
#================================================================

# This script depends on the environment variable "CMUHOME" being set, and
# the presence of $CMUHOME/os/bin (where os is sun, aix, or hpux).  These
# variables are set automatically by running $CMUHOME/etc/cshrc, profile, or
# kshrc.

#================================================================
# CHANGE THE FOLLOWING LINES:
#

## uncomment the following two lines for debugging
#exec > /tmp/shellout.$$ 2>&1
#set -x

# Connect:Direct SNODE
snode="cd_node"
```

```
# Remote User ID on SNODE, if needed
snodeid=""

# Password for ID on SNODE, if needed
snodepass=""

# destination directory on SNODE
snodedir="/home/cdunix/incoming/"

# process name--It is ok not to change this
process_name="cefa"

# $directbase is where direct lives
directbase="/home/cdunix/ndm"

# $NDMAPICFG is where the Connect:Direct api configuration lives.  You
# probably should not change this.
export NDMAPICFG=$directbase/cfg/cliapi/ndmapi.cfg


# $direct is the full path to the direct executable.  You probably should not
# change this.
direct=$directbase/bin/direct
#
# END OF CHANGE SECTION
#================================================================

#================================================================
# Usage:
#
# sample_direct.sh inputfile batch_number log_directory trans_number
#
# This script takes the name of an input file and dynamically
# generates a Connect:Direct Process to copy the file to a
# remote.  The batch_number is the batch_number generated for the file by
# the Connect:Enterprise daemon.  The log_directory is the log directory
# of the Connect:Enterprise File Agent.  The trans_number is the transaction
# number assigned to the file by the Connect:Enterprise File Agent.  These
# four arguments are configured in the fileagent.cfg file.  Without
# the batch_number/trans_number being written to the log_directory, the
# common reporting feature (cereport) will fail to operate properly.
#
# If the copy operation is successful, then the input
# file is destroyed.
#================================================================

# To submit the file to Connect:Direct, use the following function.
# Do NOT change this function to do multiple submissions.  Instead,
# call it multiple times in the body of this script.
# Takes 7 arguments:
# input_path, process_name, snode, snode_dir, snode_filename
# snode_id, snode_pass
submit_to_direct()
```

```
  SNODEID=
  if [ ! -z "$6" ]; then
    if [ ! -z "$7" ]; then
      SNODEID="snodeid=($6,$7)"
    else
      SNODEID="snodeid=($6)"
    fi
  fi
  sd_status=1
  output=/tmp/.direct_out.$$
  $($direct -x << ZEOF > $output 2>&1
  submit $2 process snode=$3 $SNODEID
    copy from (file=$1 pnode)
         to (file=$4/DIRECT.$5 snode)
  pend;
  ZEOF)
  for pnum in `cat $output|grep -i ", Process Number ="|awk '{print $NF}'`; do
    slock -b -l $FALOGDIR/direct.log
    echo $FATRNUM $pnum >> $FALOGDIR/direct.log
    slock -u $FALOGDIR/direct.log
    sd_status=0
  done
  rm -f $output
  return $sd_status

PATH=${PATH}:${CMUHOME}/bin
dir=`dirname "$1" `
filename=`basename "$1" `
inputpath=$dir/DIRECT.$filename
CEBNUM="$2"
FALOGDIR="$3"
FATRNUM="$4"

# Extract the batch into file DIRECT.$filename
cmuextract -b#${CEBNUM} -f $inputpath

exitcode=1

# Submit the file to Connect:Direct
submit_to_direct $inputpath $process_name $snode $snodedir $filename $snodeid
$snodepass

if [ "$?" = "0" ]; then
  exitcode=0
  rm -f $1
fi

exit $exitcode
```

Running CeReportCust

Before you run **CeReportCust** or **cereport** for the first time, complete all the configuration procedures in *Configuring Connect:Enterprise File Agent* on page 35. The **CeReportCust** command invokes a script that creates the **cereport** script.

1. Run **CeReportCust**.

```
$CMUHOME/etc/CeReportCust
```

   The script begins with the following message:

```
======================================================================
Sterling Commerce, Inc.,(TM) Connect:Enterprise(TM) UNIX(TM)
cereport Customization.

The customization procedure will create or override cereport script in
directory /home/enterprise/ceunix/etc.

To abort the process, enter Control-C.

======================================================================

Please press ENTER to continue...
```

2. Press **Enter** to continue.

   The following prompt is displayed:

```
$ CeReportCust

======================================================================
Sterling Commerce, Inc., (TM) Connect:Enterprise(TM) UNIX(TM)
cereport Customization.

The customization procedure will create or override cereport script in
directory /home/enterprise/ceunix/etc.

To abort the process, enter Control-C.

======================================================================

Please press ENTER to continue...

Searching your system for the Java interpreter program (java)
Searching...

******************************************************
Select the appropriate Java interpreter program.

Contact your System Administrator if you are unsure
about the appropriate selection.
******************************************************
0 = To enter the absolute path of the Java program (java):
1 = /usr/java130/bin/java
2 = /usr/java130/jre/bin/java
Select [0-2]:
```

3. Type your selection and press **Enter**.

```
Enter the FULL path name of the Connect:Direct UNIX
log directory: [/home/enterprise/cdunix/var/log]
You have chosen [/home/enterprise/cdunix/var/log] as the
Connect:Direct UNIX log directory, please confirm: [Y/n]

Enter the FULL path name of the Connect:Enterprise
base log directory [/home/enterprise/ceunix/var/log]
You have chosen /home/enterprise/ceunix/var/log as the
Connect:Enterprise UNIX log directory, please confirm: [Y/n]
-----------------------------------------------------------------
The script file /home/enterprise/ceunix/etc/cereport has been created!

To bring up the Connect:Enterprise reporting tool, run cereport at the shell
prompt.

-----------------------------------------------------------------

$ _
```

4. Accept the path name of the log directory and the script exits to the shell prompt.

Running cereport

When you run **cereport**, the requested information is displayed on the screen. For a detailed explanation of the output, refer to the *Viewing cereport Output* on page 44.

Type the following command to run cereport:

```
cereport [-h] [-?] [-f file name] [-B begin_time] [-E end_time] [-l] [-s]
```

The following table describes the **cereport** command options:

Option	Description
–h	Retrieves information on the **cereport** command and its options.
–?	Retrieves information on the **cereport** command and its options.
–f *file name*	Retrieves transaction records for a specified file name. The file name can be a canonical batch ID, or a Connect:Direct file name. In the following example, the command retrieves all transaction records for the file *tx.wfg*: cereport –f tx.wfg

Option	Description
–t *transaction_number*	Retrieves records for a particular File Agent transaction number. Records are listed in processing order. For example, if a file comes into Connect:Direct, is transferred through the File Agent, and is submitted to the Connect:Enterprise base, the records are displayed in the order “Connect:Direct, Connect:Enterprise File Agent, Connect:Enterprise base”. In the following example, the command retrieves records for File Agent transaction number 46: cereport –t 46 **Note:** Acquire the transaction number by first performing a cereport –f *file name* command.
–B *begin_time*	Retrieves a list of transaction records logged on or after *begin_time*. The format of the argument is [CC]YYMMDD[:HHMM[SS]] in 24-hour format. The default value for *begin_time* is 0, which retrieves all records in log. In the following example, the command retrieves all records logged at or after 5:00 am on October 1, 1999: cereport –B 19991001:050000
–E *end_time*	Retrieves a list of transaction records logged on or before *end_time*. The format of the parameter is [CC]YYMMDD[:HHMM[SS]] in 24-hour format. This parameter defaults to “now” which returns all records in the log. In the following example, the command retrieves all records logged at or before 12:00 pm on October 1, 1999: cereport –E 19991001:120000 Running **cereport** with both –B and –E retrieves transaction records logged between *begin_time* and *end_time*. In the following example, the command retrieves all records logged between 5:00 am and 12:00 pm: cereport –E 19991001:120000 –B 19991001:050000
–s	Retrieves transaction record and suppresses the first line column headers. This is the default display format. If the search results in no records being found, the “no records found” message is suppressed. In the following examples, the commands retrieve all transaction records for the file *tx.wfg* and display them in short format. cereport –f tx.wfg cereport –f tx.wfg –s
–l	Retrieves transaction records in long format. This option overrides the default of short format. In the following example, the command retrieves all transaction records for the file *tx.wfg* and displays them in long format: cereport –f tx.wfg –l

Viewing cereport Output

You can view different formats for output. The short format is the default report. It lists time stamp, entry point, file size, and file name. The long format lists all of the above, plus batch, process, and file modification information.

Short Format

The short format is the default for Connect:Enterprise. The sample entry and output are followed by a description of the output elements.

```
/home/server6/user1/enterprise/ceunix/etc > cereport
tran_num     action_time        ep            file_size     file_name
1            19991013:131544    U             52            add_to_mailbox.001
```

Field	Example	Description
tran_num	1	Transaction number is assigned by the File Agent and is unique for each transferred file (up to 4294967295 files).
action_time	19991013:131544	Time stamp of the transaction.
ep	U	Entry point identifies the source of the transaction. D=Connect:Direct, E=Connect:Enterprise, M=MQSeries, U=Source other than Connect:Direct or Connect:Enterprise.
file_size	5200	File size in bytes.
file_name	add_to_mailbox.001	Name of the file. Path is not included.

Long Format

You can display the long format using the **–l** option. The sample entry and output are followed by a description of the output elements.

```
/home/server6/user1/enterprise/ceunix/etc > cereport -l
tran_num     action_time        ep            file_size     file_name
             batch_num          proc_num      file_mtime
1            19991013:131544    U             52            add_to_mailbox.001
             8                  0             19991013:131545
```

Field	Example	Description
tran_num	1	Transaction number is assigned by the File Agent and is unique for each transferred file (up to 4294967295 files).
action_time	19991013:131544	Time stamp of the transaction.
ep	U	Entry point identifies the source of the transaction. D=Connect:Direct, E=Connect:Enterprise, M=MQSeries, U=Source other than Connect:Direct or Connect:Enterprise.
file_size	5200	File size in bytes.
file_name	add_to_mailbox.001	File name. Path is not included.
batch_num	8	Batch number is assigned by Connect:Enterprise. This number is not always unique.
proc_num	0	Process number is assigned by Connect:Direct. This number is not unique.
file_mtime	19991013:131545	Last modification time of the file.

Transaction Format

You can display the transaction format using the **-t** option. Three sample entries and output are provided:

```
/home/server6/user1/enterprise/ceunix/etc > cereport -t 16
20000424:105412 MQAGENTBatch Add succeeded.
  File Name = TESTWRITE1.QUEUE
  ID Used = mqm
  BID Used = user1
  Batch Number Obtained = 1293
  Transaction Number = 16

  ENTERPRISE No records show the batch has been accessed after
  creating.
```

```
/home/server6/user1/enterprise/ceunix/etc > cereport -t 5
20000407:140133 FILEAGENTRun Sync Command Succeeded.
  File Name = /sci/users/mqm/fileagent/stage/ce.1250.17
  97.user.add_to_mailbox.testfile16
  File Last Modification Time = 20000407:140048
  File Processing Time = 20000407:140133
  File Size = 1797
  Command Executed = exec /sci/users/mqm/etc/mqwrite.sh
  -d /sci/users/mqm/fileagent/control/mqwrite.trc -v -f
  /sci/users/mqm/fileagent/stage/ce.1250.1797.user.a
  dd_to_mailbox.testfile16 -t 5 -b 1250.1797.user.add_
  to_mailbox.testfile16 -l /sci/users/mqm/fileagent/control -q
  TESTWRITE1.QUEUE -r sterling.queue.manager -g yes -m text
  Command Return Code = 0
  Transaction Number = 5

20000407:140133 MQ AGENT  Successful
  FileName = /sci/users/mqm/fileagent/stage/MQW.ce.125
  0.1797.user.add_to_mailbox.testfile16
  MQSeries Name = TESTWRITE.QUEUE
  Message Size = 10000
  Processing Time = 20000407:140133
  Last Modification Time = 20000407:140134
  Mode = TEXT
  Group Messages = Y
  Group ID = 414D5120737465726C696E672E71756538E3B20C0006C02
  Bytes Written = 1747
  Messages Created = 49
  Completion Text = MQWRITE Transaction successful
  Transaction Number = 5
```

```
/home/server6/user1/enterprise/ceunix/etc > cereport -t 15
20000407:145230 DIRECT    No information was found.
                          File Name = /sci/users/mqm/fileagent/stage/add_to_mai
                          lbox.ccod.testfile19
                          File Size = 1796
                          File Last Modification Time = 20000407:145230
                          File Processing Time = 20000407:145312
                          ID Used = user
                          BID Used = add_to_mailbox.testfile19
                          Batch Number Obtained = 1259
                          Transaction Number = 15

                          ENTERPRISE No records show the batch has been accessed after
                          creating.
```

Appendix A

Regular Expressions

This appendix contains a description of how Connect:Enterprise File Agent uses regular expression substrings.

Regular Expression Substring Usage

When a file field is matched by the File Agent, the pattern can use extended capabilities of the regular expression language.

For example, a process line definition that looks like:

```
file="^(mbox[::digit::])[.](.*)" id="&1" bid="&2"
```

matches the following criteria:

1. File name must start with the explicit string **mbox**.
2. The string **mbox** must be followed by a single-digit character.
3. The single-digit character must be followed by a single period.

An implicit expression in parentheses matches the file name and is noted as &0. This value is used to replace any occurrence of &0 in a subsequent *id*, *bid*, *command*, or *synccmd* field.

The first parenthetical expression **(mbox[::digit::])** replaces a subsequent &1 value in an *id, bid, command*, or *synccmd* field.

The second parenthetical expression matches any text after the first period character, and is used to replace a subsequent &2 value in a *id*, *bid, command*, or *synccmd* field.

Caution: Use the back slash character to ignore the regular expression meaning of the character immediately following the back slash.

For example, given the configuration entry:

```
file="^(mbox[::digit::])[.](.*)" id="&1" bid="&2" \
command="rm -f %P > /dev/null 2>&1"
```

The 2>**&1** is replaced by 2>**mbox6** because the substring **&1** is special, resulting in a command of:

```
"rm -f mbox6.output.dataset 2>mbox6"
```

To prevent this replacement, use the following form instead:

```
file="^(mbox[::digit::])[.](.*)" id="&1" bid="&2" \
command="rm -f %P > /dev/null 2>\&1"
```

The back slash quotes the &1 substring and prevents the replacement. With the back slash in place, the replacement is normal:

```
rm -f mbox6.output.dataset 2>&1"
```

Index

Symbols

A

B

C

D

F