

3.0 FEMIS Notification Service

3.1 UNIX Host Notification Service

When multiple COTS applications are brought together as in FEMIS, there is the question of how they should work together. The job of the FEMIS application manager is to ensure that all the FEMIS applications can work with one another without user intervention. The inter-task Notification Service is a process for dissimilar applications to communicate with one another during operation. Applications can post and receive event notifications within the FEMIS system with the support of the Notification Service residing on the UNIX host server and on client workstations.

Each workstation hosting the FEMIS client software uses the Notification Service to coordinate activities and data at three levels. The purpose of the Notification Service is to communicate status 1) among active processes on a given workstation, 2) between workstations on the same server, and 3) among workstations on different servers. The Notification Service does not communicate data but notifies active processes of the availability and location of relevant data in a timely fashion. It is the responsibility of the interested processes to retrieve the data. Likewise, processes, which produce, manipulate, or transform data, can notify affected processes of the new state of the data.

The Notification Service also resides on the UNIX host server. Its purpose is to receive and forward notification events to other servers. Workstations connected to this server may emit notification events destined for workstations connected to other servers. These events can be forwarded between servers where the local Notification Service can determine the final destination. The UNIX host server utilizes a relational database for the organization and storage of the enterprise data. The DBMS and any other server process can also use the Notification Service to coordinate activities.

Query, manipulation, and update of data are performed by applications residing in FEMIS workstations. These applications have the responsibility to notify other applications that require the same data of any data changes. This event is communicated via the Notification Service, which serves as the single point of contact that manages the distribution of the event to relevant receivers. When necessary, the Notification Service will propagate the event to distant workstations connected to other servers.

Two parameters have been added to the `femis_event` configuration file (`femis_event.conf`). These parameters support new functionality in `femis_event`. Less comments, the two new lines are

```
com maxaux=TEMPLATE_MAXAUX
com killaux=TEMPLATE_KILLAUX
```

MAXAUX is the maximum number of AUX processes that `femis_event` will allow to be active at one time. `femis_event` monitors all AUX (ddn) processes execution and establishes a queue for AUX process requests. Whenever the number of AUX processes active reaches MAXAUX, then `femis_event` places requests in a queue and delays execution. There can be any number of queued requests; the size of the AUX process queue is dynamic.

KILLLAUX is the maximum time in seconds that an AUX process is allowed to remain active without completing. KILLLAUX seconds after execution, if an AUX process still remains active, femis_event kills it, allowing queued requests to be started.

femis_event manages the AUX process queue every 1 to 2 seconds. If an AUX process exits, the count of active processes is decremented. If there are entries in the AUX process queue, and there are fewer MAXAUX processes active, femis_event takes one off the queue and starts it. Whenever an AUX process has been requested and if there are more than MAXAUX processes active, femis_event queues the request. Otherwise, the AUX process is started immediately.

The two TEMPLATE parameters are set up during install. If not specified during install, the default values of these parameters are:

```
maxaux = 25
killaux = 480 (equal to 8 minutes)
```

Status of the AUX process queue can be viewed by using the \$i and \$proc directives in fev. Output from fev / \$i is as follows (example):

```
activeauxsize . . . . . 101
activeauxcount . . . . . 0
activeauxlimit . . . . . 7
activeauxtimeout . . . . 480
queueauxsize . . . . . 101
queueauxcount . . . . . 0
```

Parameter activeauxsize is the size of the table of currently running AUX processes. It takes on values 1, 101, 201, and so on. Parameter activeauxcount is the current number of AUX processes running. If no processes are running currently, the value is zero. Activeauxlimit is the maximum number of AUX processes allowed to be running at any one time. Activeauxtimeout is the maximum time in seconds that an AUX process will be allowed to execute. Queueauxsize is the current size of the AUX queue array. Queueauxcount is the number of AUX requests currently queued.

Output from the fev / \$proc command is a list of the following properties of the active AUX processes:

```
I           = position in array (1,2,3...)
activeauxpid = process ID number of AUX process
activeauxstart = time when process started (time_t value)
activeauxcli  = host name of client
activeauxexe  = path & file name of executable
```

3.1.1 UNIX Notification Service

This section describes the Notification Service residing on the UNIX platform, which serves as the host server. The PC version of the Notification Service is included in the installation of the FEMIS client software. Both versions have identical functions. The UNIX function that implements the

Notification Service is called `femis_event`. The purpose of `femis_event` is to provide PC users of the FEMIS event notification system a communication path for the sharing of event information with each other. Events posted at one PC are sent to other PCs on the network by communicating with one or more notification servers.

Local events posted at one PC client workstation are received at the notification server running on the LAN and then sent out to all clients that have expressed an interest in that event. Global events posted at one PC client workstation are received at the notification server running on the LAN and then sent out to clients on that LAN and also to other notification servers on the WAN.

The `femis_event` is normally run as a background daemon process. Scripts that are used to startup the FEMIS system also invoke the notification server.

As do all sockets servers, `femis_event` utilizes a predefined service port on which to listen for client connection requests. By default, the service port is obtained from a definition in `/etc/services`, the standard UNIX data file of Internet services and aliases. The standard service name of the notification server is `femis-notify`.

3.1.1.1 Executable Binary Files

Two executable binary files are in the UNIX notification subsystem.

```
/home/femis/bin/femis_event : notification server executable  
/home/femis/bin/fev : a client application for UNIX environment
```

3.1.1.2 Configuration Data File

The notification server utilizes one configuration file.

```
/home/femis/etc/femis_event.conf : notification server configurations
```

3.1.1.3 Service Port Data File

The three FEMIS network protocols multiplex on service port 1776. A definition of `FEMIS 1776` must be present in the UNIX service ports data file (`/etc/services`).

3.1.1.4 Protocol Numbers

The current FEMIS protocol numbers are identical to the legacy FEMIS service port numbers. Including the obsolete meteorological protocol, the names and numbers are as follows:

9015	<code>femis-command</code>	command server daemon
9020-35	<code>femis-notify</code>	notification service
9037	<code>femis-metdata</code>	meteorological data daemon
9040	<code>femis-monitor</code>	femis monitor daemon

In the event of service port or protocol number conflicts, contact PNNL immediately before attempting to reconfigure the IP port addresses, which must be performed before a correct installation of the FEMIS network daemons can be accomplished.

3.1.1.5 Daemon Server Startup

Scripts should be used to start or restart the notification server daemon. The following script will successfully start and restart the command and notification servers:

```
# sh /etc/init.d/femis {start or stop}
```

To stop and start `femis_event` from command line, you should use the `stopnotify` and `startnotify` scripts in `/home/femis/bin`. The `femis` script in `init.d` is a specialized automated script and may cause adverse side effects if run manually from the command line.

3.1.2 Notification Server Configuration Options

3.1.2.1 Command-line Options

The command-line options of program `femis_event` that are defined in this section are

```
femis_event          : executes in foreground
femis_event -c       : executes a clone in background
femis_event -H homedir : specifies path to FEMIS home directory
femis_event -v       : report the current version
femis_event -V       : report the current + rcs versions
femis_event -q       : quiet mode
femis_event -Q       : really quiet mode
femis_event -d       : executes with many diagnostics
femis_event -a       : enable keep alive mode
femis_event -q -d    : executes with only a few diagnostics
femis_event -L FFFF  : write a verbose log file named FFFF
femis_event -l FFFF  : write a brief log file named FFFF
femis_event -e FFFF  : write an error only log file
femis_event -s SSSS  : specifies service name for getservbyname
femis_event -S       : uses service name femis-notify if found
femis_event -p PPPP  : gets port number from environment variable PPPF
femis_event -t secs  : RESERVED - NOT IMPLEMENTED (see note)
femis_event -i       : report primary ip address and port number
femis_event nnnn     : use port nnnn instead of standard
femis_event host     : connect to named server
femis_event host host : connect to named servers (see note)
femis_event -r       : use registered service port (1776)
femis_event -conf file : specify a configuration file path/name
femis_event # host host : port number # and a list of hosts
femis_event -u       : use unregistered service port (9020-29)
```

Normally, only `femis_event -c host` will be needed to start executing a notification server. However, the additional options can be mixed to provide logging, diagnostics, and nonstandard service port usage.

3.1.2.2 Clone Process in Background Option

When this option has been included anywhere on the command line, the `femis_event` program clones itself and then the parent exits, leaving the child process to carry on as a background daemon process.

```
if (fork () != 0)
  exit (0);
....
```

Example: `femis_event -c`

3.1.2.3 Display Version Options

Including `-v` or `-V` anywhere on the command line with `femis_event`, causes the current version or the current version with RCS version to be displayed. Example:

```
% femis_event -v
FEMIS_EVENT - Version 1.0.11 - Wed Dec 14 15:19:49 PST 1994
% femis_event -V
FEMIS_EVENT - Version 1.0.11 - Wed Dec 14 15:19:49 PST 1994
  Copyright © 1994 Battelle Memorial Institute. All Rights Reserved.
RCS: $Id: femis_event.cc,v 1.2 1994/12/14 23:17:08 d31033 Exp d31033$
```

The `femis_event` version is the current code version, not the FEMIS nor the RCS version. The date and time indicate when the executable was compiled and linked.

3.1.2.4 Diagnostic and Quiet Modes

Using `-d` causes diagnostics to be printed out when running in foreground mode, i.e., not using option `-c`. Including `-q` or `-Q` with `-d` limits the amount of diagnostic information printed out. Options `-q` and `-Q` mean quiet and real quiet respectively. Using `-d` alone produces verbose diagnostics. Using `-d -q` limits the diagnostics. Using `-d -Q` limits all but severe diagnostics. Examples:

```
% femis_event -q : quiet mode
% femis_event -Q : really quiet mode
% femis_event -d : executes with many diagnostics
% femis_event -q -d : executes with only a few diagnostics
```

3.1.2.5 Service Port Name Option

Including this option lets you specify the service port name on the command line rather than using the default name, `femis_notify`. Example:

```
% femis_event -c -s evtserve-test-3-eoc
```

For this command to work correctly, the service name `evtserve-test-3-eoc` must have been entered in the `/etc/services` data file.

Using option `-s` causes the standard service port name to be invoked.

3.1.2.6 Service Port Environment Option

This option lets you specify service ports in environment variables. Example:

```
% setenv MY_FEV_PORT 9027
% femis_event -p MY_FEV_PORT -c
```

3.1.2.7 Display IP Address and Service Port

When the notification server is started with the `-i` option, rather than starting up a Notification Service, it just reports status information about network addresses and then exits. Information displayed includes the date/time of the last build (version identification), name of the local host, primary IP address of the local host, and service port number for the client connections. Example:

```
> su - femis
Password: *****
> femis_event -i
Last build ..... Thu Oct 17 11:54:08 PDT 1996
Host name is ..... fallout.pnl.gov
IP address is .... 130.20.92.118
Port number is ... 9020
>
```

The purpose of this directive is to obtain information needed in the multiple IP address workaround. Also see Section 2.10.3 Setting Up `femis_event`, in the *Installation Guide for FEMIS Version 1.5*.

3.1.2.8 Enable Log Files

These options let you enable log file output from `femis_event`. The `-e` option creates an errors-only log file. Option `-l` produces a brief diagnostic log file. Option `-L` generates a verbose log. Place the desired file name in the argument following `-e`, `-l`, or `-L`. Examples:

```
% femis_event -e errors-only.log.12-24-94 -c
% femis_event -L femis_event.log.12-25-94 -c -p XMAS_PORT
% femis_event -l /home/femis/log/femis_event.log`date +%Y%m%d.%H%M`
```

3.1.2.9 Nonstandard Port from Command Line

The notification server can be started with a nonstandard service port without the need for changes in `/etc/services` (which requires root access) or changing the environment variables simply by including the desired port number on the command line (specify only once). Example:

```
% femis_event -c 9920
% fev - 9920
```

3.1.2.10 Connecting to Other EOC's Notification Server

To have the notification servers at multiple EOCs connected together, include the names of the other EOC server hosts on the command line. Example:

```
server1:% femis_event -c server2
server2:% femis_event -c server1
```

3.1.2.11 Multiple Remote EOC Servers Limitation

For this release, no special server-to-server algorithms for routing have been implemented in the notification server. Smart routing algorithms may be implemented in a future version. Also, the `-t` option, a part of multi-host, is not implemented.

If you specify only one remote host, you get the optimal routing, which is host-to-host with no alternate conditions or routes.

If you specify two or more remote hosts, the local server connects with all the remote hosts you named. Global event messages are then relayed to all specified remote hosts, even though that may not be necessary. As a result, global messages may be sent to a remote host more than once.

3.1.2.12 Server to Server Connection

The FEMIS UNIX notification server (`femis_event`) supports a network of multiple notification servers. Any number of server programs can interconnect with each other, and the purpose of this interconnection is to provide a medium for communicating global event messages, provided that topology of the network is not a concern.

To establish connection to other servers, a list of notification servers can be included on the command line. The syntax to designate a notification server connection is as follows:

```
host name (uses default service port)
```

In the following lines, all servers use the same default service port number. Example:

```
%femis_event -c countyeoc stateeoc
%femis_event -c irzcountyeoc pazcountyeoc stateeoc
```

Multiple notification servers can be executed on the same host by using a different service port number for each instance. The syntax to designate multiple notification server connections is as follows:

```
%<port number>@<host name>    port-and-host using registered service port
%<port number>#<host name>    port-and-host using unregistered service port
```

At the current time, only the registered service port method is being utilized by FEMIS systems fielded by CSEPP.

In the following lines, two notification servers are started and each is cross connected to the other. Example:

```
%thiseoc:/home/femis/exe/% femis_event -c 9021 9022@thiseoc
%thiseoc:/home/femis/exe/% femis_event -c 9022 9021@thiseoc
```

In the above example, unregistered service ports 9021 and 9022 are used rather than the default service port 9020. Server 9021 is connected to server 9022, and server 9022 is connected to server 9021. These connections are on the same host.

In the current FEMIS release, both concepts above have limitations. First, event routing is not optimized for more than two notification servers. Thus, a single event declaration will be sent multiple times on inter-network links.

A network of notification servers can be started by implementing exact topology in a series of startup commands. Example:

```
posteoc% femis_event -c 9020 9020@countyeoc 9020@stateeoc
countyeoc% femis_event -c 9020 9020@posteoc 9020@stateeoc
stateeoc% femis_event -c 9020 9020@posteoc 9020@countyeoc
```

The above example starts notification servers on three hosts: posteoc, countyeoc, and stateeoc. Each is capable of sending global event messages to the other two. No regard is given to topology, i.e., each server sends events to the other two servers, even if having one of the others do a relay would accommodate more efficient use of network bandwidth.

An alternate way to start the servers is to start one, then add one to the network, and later add the third. Example:

```
posteoc% femis_event -c 9020
```

The above established a single notification server. Next enter:

```
countyeoc% femis_event -c 9020 9020@posteoc
```

We now have a two-node event server network: `countyeoc` connects to `posteoc`, which learns of the new server-to-server connection. Next enter:

```
stateeoc% femis_event -c 9020 9020@posteoc 9020@countyeoc
```

We now have a three-node event server network. `Stateeoc` connects to both `posteoc` and `countyeoc` and each learn of the new server node.

Graceful removal of nodes from the notification server topology and optimization of topology for saving network bandwidth have not yet been implemented. These will be done in future FEMIS releases.

3.1.2.13 Which Service Port to Use

Which service port the notification server uses is determined as follows: from the following list, the first service port that produces a valid service port number is used as the service port method for this daemon server.

- If the port number is included on the command line, then that port is used even if the methods below also produce a valid service port number. Example:

```
femis_event 9975
```

- If a service name is included on the command line (via `-s` or `-S`), then that service name is used in a `getservbyname()` call. If that service name returns a valid service port from the `/etc/services` data file, then that port is used. Example:

```
femis_event -s FEMIS_ShellServer
```

- If an environment name is included on the command line, then that environment name is translated into a service port number. Example:

```
setenv MYPORT 7120 ; femis_event -p MYPORT
```

- The default service name, `femis-notify`, is tried in a call to `getservbyname()`. If that returns a valid service port, then that port number is used.
- The default environment name `FEMIS_EVENT_PORT` is translated. If that name is defined and translates to a valid port number, then that service port is used.
- If all the above fail, `femis_event` terminates with an error.

Normally, you can just use the standard service port number from the `/etc/services` file. However, for testing and diagnostics, additional methods have been included for running additional notification server modules that use a nonstandard port number, so there is no interference with normal operations.

3.1.2.14 Enable Keep Alive

If the UNIX notification server is started with `-a` specified, keep alive mode for all socket calls is utilized.

3.1.2.15 Registered and Unregistered Service Port

Command line option `-r` specifies use of the registered service port only. Command line option `-u` specifies use of the unregistered service ports only. The default starting is the registered service port. Previously the default was to unregistered ports. For more information, see Section 6.0, FEMIS Contact Daemon.

Whether the `femis_event` was executed using `-r` (registered and default) or `-u` (unregistered) method, both methods are able to cross connect with other `femis_event/s` that can be of either type. However, the `startnotify` script must know which method to utilize. For registered, use `PORT@HOST`. For unregistered, use `PORT#HOST`.

3.1.3 femis_event EVENT Configuration File

The `femis_event` uses a configuration file. The default `femis_event` configuration file is located at `/home/femis/etc/femis_event.conf`. This configuration file contains set up information and details of command line options for auxiliary processes, e.g., DDN and DEI scripts.

Auxiliary `femis_event` processes are utilized by the FEMIS Data Driven Notification (DDN) and DEI scripts. DDN processes are Perl scripts. See Section 7.0, FEMIS Data Exchange Interface (DEI), for more information on DEI.

To specify a `femis_event` configuration file path/name other than the default, use the `-conf <file>` command line option to `femis_event`.

The configuration file is a plain text file. Parsing rules are as follows:

- Any line starting with a `#` is a comment line.
- The line `com port=registered` specifies the registered service port to be used when no command line option is specified. Command line options `-r` and `-u` override this command.
- The line `com port=unregistered` specifies the unregistered service port to be used when no command line option is specified. Command line options `-r` and `-u` override this command.

- The line `com fevpath=femisbin` specifies to look in `/home/femis/bin` for the `fev` executable.
- The line `com fevpath=dotslash` specifies to look in `./` for the `fev` executable.
- A line starting with `aux` specifies information pertaining to the launching of auxiliary processes.
- `aux argname=on` turns argument naming on. In this mode, arguments to the auxiliary process are passed as `-<name> <value>`. If `aux argname=off` is specified, arguments are passed just as `<value>` with no argument naming utilized. Naming allows for free format argument lists.
- `aux keypos=ITEM` specifies the position of which item to key on. Possible ITEMS are `msgname`, `exerid`, `auxprocessid`, and `parm#`. The `keypos` option specifies which message field becomes the key field for selection of an auxiliary process to be launched.
- `aux ifport=PORT` specifies only launch this command if the notification server's port/protocol is equal to `PORT`. `PORT` is a decimal number value. If this option is not specified, the command is always launched. If the option is present and `PORT` is not the port/protocol, the command will not be launched.
- `aux notport=PORT` specifies only launch this command if the notification server's port/protocol is not equal to `PORT`. `PORT` is a decimal number value. If this option is not specified, the command is always launched. If the option is present and `PORT` is equal to the port/protocol, the command will be launched.
- `aux exe=path/file` specifies the path/file name of the auxiliary process executable file. The file must be tagged as `x` (executable) in the file system. The executable file can be a compiled/linked program, a shell script, a Perl script, or any executable.
- `aux key=VALUE` specifies what value the key field must be equal to in order to select and launch this command.
- `aux arg=ITEM` specifies an item to include in the argument list to the auxiliary process. The possible ITEM names are `msgname`, `exerid`, `auxprocessid`, `parm#`, `origin`, `msgflags`, `message`, `home`, `host`, `port`, `stdport`, and `fev`.

All ITEMS are extracted from the `<...message...>`. ITEMS are as follows: `MsgName` is message name. `ExerID` is the exercise identification. `AuxProcessID` is the auxiliary process identification. `Parm#` is parameter number. `Origin` is the complete origin string from the originating PC notification code. `MsgFlags` is the message flags, bit encoded. `Message` is the full and complete message string. `Home` is the `femis_event` home directory, e.g., `/files13/home/femis`. `Host` is the server's host name. `Port` is the port/protocol number, e.g., `9020`. `StdPort` is `Yes` or `No` depending on whether standard service port (`1776`) is in affect. `fev` is the complete string for launching `fev`, for use in the auxiliary process, including path, name, and port number.

3.1.4 Notification Server Utilities

3.1.4.1 UNIX Client Application – fev

The notification server subsystem includes a client for the UNIX system environment. The UNIX client can be used to test features of the command server, both new and old, and to perform certain diagnostics.

Note: This client is not an integral FEMIS system component.

The file name of the UNIX client is `fev`. The UNIX client is installed at the same subdirectory as the notification server (see Section 3.1.1.1, Executable Binary Files).

In addition to testing, `fev` is also used in FEMIS DDN, DEI, and AutoRecovery.

3.1.4.2 UNIX Client Command-line Options

Valid command-line options for `fev` have a format and usage similar to those for the notification server. Example:

```
% fev host nnnn -- nonstandard port and host from command
% fev - nnnn # nonstandard port local host (testing only)
% fev -p PPPP # nonstandard port from environment variable
% fev -s SSSS # nonstandard port from /etc/services file
% fev -S # use standard service name femis-notify
% fev -i IDNUM # specify notification client id number
% fev -x # don't exit immediately on eof from standard-input
% fev -u # use unregistered service port (9020-29)
% fev -r # use registered service port (1776)
% fev -f: connect to femis_event using FIFO for diagnostic use
% fev -d: diagnostics enabled
% fev -H: HOMEDIR set path of /home/femis
% fev NUMBER@HOST - connect to Number on Host using registered service port.
% fev NUMBER#HOST - connect to Number on Host using unregistered service port.
```

See descriptions of these options in Section 3.1.2, Notification Server Configuration Options.

3.1.4.3 Client ID Number

You can simulate what happens when a notification system client crashes and then comes back online. In that case, the PC/client needs to receive the same client ID number that was assigned to that PC/client during the previous session. The notification server handles that scenario correctly, but during testing on a single development host, you need to tell the UNIX client which PC/client is connecting by specifying the PC/client ID from the previous session (see `o` command reply in the following sections).

Syntax: `fev -i IDNUM`

3.1.4.4 UNIX Client Protocol

To run the notification server UNIX client, do the following:

```
% set path = (/home/femis/exe $path)
% fev # connect to local host, standard port
% fev <remote host> # connect to a remote host
% fev - <port> # connect to nonstandard port on this host
% fev <remote host> <port># connect to nonstandard port on remote host
```

The notification service UNIX client provides several shorthand commands to the actual notification server protocol, as follows:

```
o : sends open-link message (NS_MT_OPENLINK)
    reply message contains the client's link id
c : sends close-link message (NS_MT_CLOSELINK)
i EEEE : sends register-interest message (NS_MT_REGISTER_INTEREST)
r EEEE : sends remove-interest message (NS_MT_REMOVE_INTEREST)
e EEEE : sends declare-event message (NS_MT_EVENTMSG) (nonglobal)
g EEEE : sends declare-global message (NS_MT_EVENTMSG & NS_EF_GLOBAL)
t1 : bombard server with multiple NS_MT_EVENT testing
t2 : bombard server with multiple NS_MT_EVENT testing
```

3.1.4.5 UNIX Client Example

Example:

```
server1:% femis_event -c 9020@server2
FEMIS_EVENT port is 9020
server2:% femis_event -c 9020@server1
FEMIS_EVENT port is 9020
server3:$ fev server1 9020
FEMIS_EVENT port is 9020
o
<<<<<< received OPENLINK-reply: client-id = 2
I TestEvent
I GlobalEvent
server4:>%fev server1 9020
FEMIS_EVENT port is 9020
o
<<<<<< received OPENLINK-reply: ...
client-id = 3
e TestEvent

<<<<<< received notification: event=TestEvent

c
^D
server4:% fev server2 9020
o
<<<<<< received OPEN-LINK-reply: ...
client-id = 2
```

```
e TestEvent
g GlobalEvent

<<<<<< received notification: event=GlobalEvent

c
^D
c
^D
```

In the example, the operator runs the notification server on two hosts, `server1` and `server2`; they connect to and communicate with each other because the other's port at host is on the command line.

Next, the client is run on `server3`, connecting to `server1`, a link is opened, and interest is declared in two events, `TestEvent` and `GlobalEvent`. Also, the client is run on `server4`, connecting to `server1`, a link is opened, and event `TestEvent` is declared. Because the client on `server3` has declared interest, a notification message is delivered and reported there.

The client on `server4` is next terminated (via close link and `control-D`). The `server4` client is rerun, this time connecting to `server2`, and the link is opened. The event `TestEvent` is then declared at `server2`. Nothing happens at `server3`, as `TestEvent` is local (not global) to the server on `server2`.

Finally, the client on `server4` declares a global event (`GlobalEvent`), and the client on `server3` is notified.

Both UNIX clients are then terminated via close link and `Control-D`.

3.1.4.6 UNIX Client Diagnostics

The UNIX client `fev` has features whereby it can spy on what notification servers are doing and what the status of each connection is. The commands are

```
$i : sends back information and statistics
$s : sends back socket connections information
$aux : sends back auxiliary socket connection information
$rem : sends back remote server list
$eve : sends back listing of server's event board
```

3.1.4.7 UNIX Client Information Diagnostic \$i

Entering `$i` at the `fev` UNIX client's terminal causes statistics information to be returned to the client. Example:

```
% fev server1
FEMIS_EVENT port is 9020
$i
FEMIS_EVENT - Version 1.0.11 - Wed Dec 14 15:54:18 PST 1994
started time . . . . . Sat Dec 17 03:00:09 1994
current time . . . . . Mon Dec 19 13:51:59 1994
pid . . . . . 23473
ppid . . . . . 1
uid . . . . . 30508
gid . . . . . 30508
dir . . . . . /home/femis/exe
home . . . . . /home/femis/sunos/home/femisuser
home directory . . . . /files8/home/femis
etc directory . . . . . /files8/home/femis/etc
log directory . . . . . /files8/home/femis/log
config file name . . . /home/femis/etc/femis_event.log
log file name . . . . . <Null>
host . . . . . server1
operating system . . SOLARIS
port . . . . . 9020
background . . . . . Yes
accepts . . . . . 192
connects . . . . . 1
reconnects . . . . . 0
messages rcvd . . . . 11826
characters rcvd . . . . 513556
messages sent . . . . 1274
characters sent . . . . 85600
malloc arena/used . . 61448 35416
evtbuf cur/tot/peak . . 2 9 9
evtbrd cur/tot/peak . . 2 9 2
intl1st cur/tot/peak . . . 288 2607 306
```

From the display above, you know the following information about the notification server daemon: has been up for 2 days, was started at 3:00 a.m. on Dec 17, is the Dec 14 version; the process ID is 23473; the sever is in background (because ppid == 1); its uid is 30508 (femis account); user's home is /home/femis/sunos/home/femisuser; the host's name is server1; the service port number is 9020 (the standard port); the notification server is running as a clone in background; and the server currently has 35416 bytes of dynamic memory allocated.

Furthermore, the server has accepted 192 connections, has established one connection itself (to the other server), has done no reconnects (because of connection termination), has received 11826 messages containing a total of 513556 characters, and has transmitted 1274 messages containing a total of 85600 characters. Using either received or transmitted, the average message length is approximately 42 characters.

For event library statistics evtbuf, evtbrd, and intl1st, also reported are current, total, and peak.

Character and message counts utilized in the diagnostic messages overhead are not included in the totals displayed.

3.1.4.8 UNIX Client Socket Connections Diagnostic \$s

Entering \$s at the fev UNIX client's keyboard causes socket connection information to be sent to the UNIX client's display. Example:

```
% fev server1
FEMIS_EVENT port is 9020
$s
```

The heading of the display contains the following:

```
ii : index number in femis_event's internal database
lism : 1 if socket is the server's primary listening socket
acpt : 1 if connection was accept()-ed on this socket
conn : 1 if connect() was established on this socket
stio : 1 if this is one of the standard I/o files
svrc : 1 if accept or connect is to another server
chan : the channel number
iana : 1 if using standard IANA registered service port for connection
host : name of the host to which this socket is connected
IP : the IP address to which this socket is connected
hwid : 32 bit hardware id number - derived from IP address
anid : the notification system client id number
when : when (date and time) when connection was established
rcv : number of messages and number of characters received
xmt : number of messages and number of characters transmitted
```

Example display of first 12 parameters:

```
ii lism acpt conn stio svrc chan iana : host : IP : hwid : anid :
3 1 0 0 0 3 1 : server1.pnl.gov : 130.20.76.45 : 82144C2D : 0 :
4 0 1 0 0 0 4 1 : server5.pnl.gov : 130.20.28.29 : 82141C1D : 19 :
5 0 1 0 0 1 5 1 : server2.pnl.gov : 130.20.242.31 : 8214F21F : 0 :
6 0 1 0 0 0 6 0 : 130.20.28.131 : 130.20.28.131 : 82141C83 : 71 :
7 0 1 0 0 0 7 1 : server6.pnl.gov : 130.20.60.103 : 82143C67 : 47 :
8 0 1 0 0 0 8 1 : server4.pnl.gov : 130.20.92.71 : 82145C47 : 69 :
9 0 1 0 0 0 9 1 : server3.pnl.gov : 130.20.92.87 : 82145C57 : 0 :
10 0 1 0 0 0 11 1 : server7.pnl.gov : 130.20.92.39 : 82145C27 : 53 :
```

Example display of final 5 parameters:

```
when : rcv : xmt
Sat Dec 17 03:00:12 1994 : r 0 0 : x 0 0
Mon Dec 19 09:50:29 1994 : r 255 11115 : x 7 473
Sat Dec 17 03:00:24 1994 : r 0 0 : x 4 319
Mon Dec 19 10:47:17 1994 : r 91 3896 : x 8 547
Mon Dec 19 10:27:49 1994 : r 259 11303 : x 8 547
Mon Dec 19 10:45:24 1994 : r 56 2335 : x 2 117
Mon Dec 19 11:14:17 1994 : r 13 13 : x 0 0
Mon Dec 19 10:29:36 1994 : r 56 2335 : x 2 117
```

From the above display, we can say that five clients currently have active connections, that client ID numbers range from 19 to 71, and that one client has no entry in the local name table (IP address 130.20.28.131).

Socket 3 is the listening socket. Socket 5 connects to the notification server on `server2`. Socket 9 is the client doing diagnostics.

Character and message counts utilized in the diagnostic messages are not included in the totals displayed.

3.1.4.9 UNIX Client Auxiliary Connect Information Diagnostic `$aux`

Entering `$aux` at the `fev` UNIX client keyboard causes the auxiliary connect information to be sent to the UNIX client's display. Example:

```
% fev server1
FEMIS_EVENT port is 9020
$aux
```

The heading of the display that follows contains

```
ii : index number in femis_event's internal database
conn : connect mode = L C A
svrc : server circuit = 0 1
auxtype: aux connection type S C U
host : name of host to which this socket is connected
hwid : 32 bit hardware id number - derived from IP address
port : port number of server/client at remote end
pid : process id number of server/client process at remote end
cid : client id number of server/client process at remote end
```

Example listing:

```
5 L 0 : U : virus.pnl.gov : 8214F20A : 9020 : 14415 : 0
6 C 1 : S : locusts.pnl.gov : 8214F20B : 9020 : 12093 : 0
7 A 0 : U : : 0 : 0 : 0 : 46
8 C 1 : S : temblor.pnl.gov : 8214F20C : 9020 : 19831 : 0
9 A 0 : U : : 0 : 0 : 0 : 38
10 A 0 : U : : 0 : 0 : 0 : 48
11 A 0 : U : : 0 : 0 : 0 : 43
12 A 0 : C : hattrick : 82145C57 : 9020 : 2593 : 0
```

3.1.4.10 UNIX Client Remote Servers Diagnostic `$rem`

Entering `$rem` at the `fev` UNIX client keyboard causes the remote connect information to be sent to the UNIX client's display. Example:

```
% fev server1
FEMIS_EVENT port is 9020
$rem
```

The heading of the display that follows contains

```
RemoteServer : Port number @ host name of the remote notification server
IPAddress : IP address of the remote host
Address : 32 bit hardware id number - derived from IP address
```

Example listing:

```
RemoteServer : IPAddress : Address
9022@virus.pnl.gov : 130.20.242.10 : 8214F20A
9021@temblor.pnl.gov : 130.20.242.12 : 8214F20C
```

3.1.4.11 UNIX Client Event Board Diagnostic \$eve

Entering \$eve at the fev UNIX client keyboard causes the server's event board information to be sent to the UNIX client's display. Example:

```
fev - test client for femis_event server
FEMIS_EVENT port is 9020
$eve
```

The heading of the display that follows contains

```
EventName : name of the event
ExerID : exercise id
Par1 : first parameter
Par2 : second parameter
Par3 : third parameter
GMT : date/time event declared
RecID : record id
```

Example listing (abbreviated):

MsgName	: ExerID	: Parm1	: Parm2	: Parm3	: GMT	: RecID
CSEPPEvent	: 0	: 10000299	:	: ALL_OVER	: 18:25	: 37
MD2	: 1295	: Operations	:	: UPD:10001	: 18:38	: 41
PLN:PlanChanged	: 0	: 10000107	:	:	: 18:17	: 33
PLN:TaskChanged	: 0	: 10000006	:	: 21	: 16:17	: 23
RSB:EventLogAdd	: 0	: J	: AckEvent	:	: 18:25	: 39
RSB:EventLogAdd	: 1295	: J	: D2:10001	:	: 18:37	: 40
Udept	: 0	:	:	:	: 15:19	: 19
Ufacility	: 0	:	:	:	: 15:16	: 18
UlocalID	: 0	: TEADTEAD	: alstuff	:	: 15:48	: 43

3.1.4.12 UNIX Client Synchronize Action \$sync

Entering \$sync and a qualifier at the fev client keyboard causes the server to send the same message back to fev, which can utilize reception of known dollar-sync messages to synchronize certain events and actions.

The UNIX client uses the command \$sync exit to synchronize forced exit while running in script mode, which must be used in conjunction with the -x option.

Example script:

```
#!/bin/csh -f
#
fev -x virus 9020<<eod
o
g My-Event 1 "par one" par_two par3
g My-Event 123 "" - 999.000
g Your-Event 99 - - -
c
\$sync exit
eod
```

The above script runs fev, opens a link, declares the three events, closes the link, and synchronizes a forced exit.

3.1.4.13 Data Driven Notification Command Line Arguments

A Data Driven Notification (DDN) command line interface has been added to fev. This feature now allows a single event including DDN parameters to be constructed and sent by fev, based solely on new command line arguments. The presence of DDN command line arguments signals fev to utilize single event mode, instead of entering interactive mode.

The following are DDN command line arguments for fev:

Argument	Function
-global	This is a global event
-nopost	Do not post this event
-aux	Launch an auxiliary process
-host HOST	Name of host to receive this event
-port PORT#	Port # or protocol # to receive this event
-dest PORT@HOST	Number and host to receive this event
-dest PORT#HOST	Number and host to receive this event
-msgname MSG	Message name
-msgflags FLAGS	Message flags

-origin ORIGIN	Origin field
-exerid EXERID	Exercise ID
-auxprocessidnet AUXID	Auxiliary process id (in femis_event.conf)
-parm## PARM##	Parameter no. ## (up to 50)

3.2 PC Notification Service

3.2.1 PC Notification Service Overview

This section describes the PC Notification Service, which serves as the PC workstation component of the FEMIS Notification Service. The PC Notification Service is designed to provide a path for sharing notification information between PC applications, PC workstations, and UNIX notification servers. Events posted by applications within a PC workstation are first sent to all notification clients on that PC, then forwarded to a UNIX notification server for distribution to other workstations and other notification servers.

The PC Notification Service operates in the background and provides services to PC applications through function calls and window messages. There is no direct user interface except the Notification Service log window, which displays diagnostic messages as the service is running.

The PC Notification Service is implemented as a stand-alone service and is automatically activated when client applications are started and remains active until all clients have been closed. There are no separate startup or shutdown procedures. Instead, notification startup and operations are controlled through configuration files and client function calls, not through command-line options.

3.2.1.1 Executable Binary Files

The PC Notification Service has two executable binary files:

```
FNOTIFSV.EXE      Notification Service executable
FNOTIF32.DLL     Notification Service function library
```

These files are normally located in the WINNT\SYSTEM32 directory but may be placed elsewhere, as long as they can be found on the system search path.

3.2.1.2 Notification Service Startup

Since the Notification Service is started by the Notification Service DLL, the user has no control over startup operations. Instead, startup parameters are read from a configuration file and can be adjusted to suit the needs of a particular installation.

3.2.2 PC Notification Service Configuration Options

The PC Notification Service can be customized by modifying one or more configuration parameters. These parameters allow you to change Notification Service behavior to accommodate client needs and special requirements. For instance, a remote user connected via a modem may need to increase the timeout limit for notification server connections, or a stand-alone installation might want to disable all network monitoring. Each of these requirements can be satisfied by adjusting the configuration parameters to fit the client's needs.

3.2.2.1 Configuration Parameters

Each configuration parameter has a unique name and most have a default value. The available configuration parameters are as follows:

Parameter	Purpose	Default Value
RunAsStandAlone	StandAlone flag (True/False)	False
SocketMaxWait	Socket timeout value (seconds)	10
LostConnCheckInterval	Lost connection check (seconds)	30
LostConnRetryInterval	Lost connection retry (seconds)	30
EventQueueSweepInterval	Queue sweep interval (seconds)	1
DefaultNotifServerHost	Default server host name	none
DefaultNotifServerPort	Default server port	none

If the default value for a parameter is not satisfactory, you can assign a more suitable value. However, you must be careful that the new value is reasonable and does not have an adverse effect on Notification Service operation.

3.2.2.2 Notification Service Configuration File

Notification Service configuration parameters are specified in a configuration file, `FEMIS.INI`, usually located in the Windows home directory. Each configuration parameter is specified by a key and its associated value, grouped under the `[Notification Service]` section.

A typical `INI` file might look like this:

```
[Notification Service]
;---Notification configuration parameters---
;RunAsStandAlone = False
LostConnCheckInterval = 10
LostConnRetryInterval = 60
```

To create an entry for a configuration parameter, insert a new line that specifies the parameter's name and its new value, separated by an equals sign (=). Key names are not case sensitive, and all blank padding is ignored.

To disable an entry, put a semicolon as the first non-blank character in the entry, which causes the line to be treated as a comment and ignored in all parameter processing.

3.2.2.3 Command-line Options

The PC Notification Service has no command-line options.

3.2.2.4 Environment Variables

No environment variables are used by the PC Notification Service.

3.2.2.5 Host Server Name and Port

UNIX host server name and port number are set by client function calls and are not directly controlled by configuration options. However, clients can use the `DefaultNotifServerHost` and `DefaultNotifServerPort` configuration parameters to store server identification information.

Note: FEMIS does not support concurrent connections to multiple notification servers. Only one server can be connected at a time.

3.2.3 PC Notification Service Operation

Operation of the PC Notification Service is discussed in the following sections.

3.2.3.1 Notification Service Window

The Notification Service window enables a user or System Administrator to view information about notification system operations. This window provides information about the system status and current version, along with a log of recent diagnostic messages.

Local Notification is what is running on the PC. The FEMIS Notification Service (PC) icon on the Windows task bar indicates current status, which will be one of the following:

- Stand-alone - blue icon with green border
- Connected to server - blue icon with black border
- Lost connection - blue icon with red border and a red slash across it

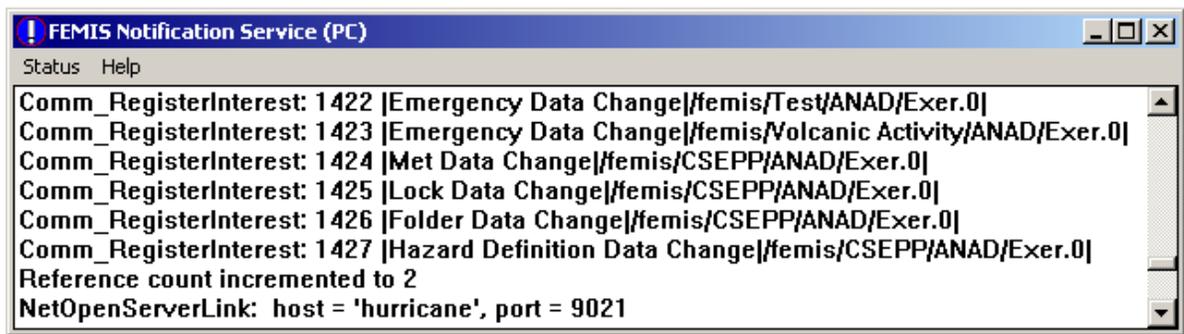
Note: If the icon is blue, has a red border, and does not have a red slash across it, then no server has been selected and notification is not shared with other PCs.

The Notification server response time from your PC can be checked by completing the following steps:

1. Open the FEMIS Notification Service (PC) window.
2. Select `Status` → `Server Response Time`. If you are connected to a server, a popup message box will display the current response time.

If the connection is very slow, it may take up to 30 seconds to determine the response time. The message window automatically closes itself after one minute.

Figure 3.1. FEMIS Notification Service Window



The `Notification Status` window displays information about local and server status, client count, event count, server host name, and server port number. The `Notification Status` window updates itself automatically.

For version information, select `Help` → `About` on the FEMIS Workbench. The `About` window will display, which contains version and copyright information.

For diagnostic information, consult the main `Notification Service` window. This window displays recent diagnostic and error messages, including network messages to and from the server and attempts to restore lost server connections.

3.2.3.2 Lost Connections

Lost connections with the UNIX notification server are a common problem and occur for a variety of reasons. The PC Notification Service is designed to automatically detect and restore lost connections, with minimal impact on FEMIS software operations.

Whenever a lost server connection is detected, the PC Notification Service sends a diagnostic message to the log window, activates the Lost Connection icon, and goes into restoration mode. Every few seconds, as specified by the `LostConnRetryInterval` value, the Notification Service attempts to contact the server and restore the connection. During this time, local notification still

occurs, but all messages to and from the server are lost and cannot be recovered. When the server finally answers, the connection is restored and the Notification Service returns to normal operation.

As discussed in Section 3.2.3.1, Notification Service Window, you can use the status icon or status window to monitor lost connections.

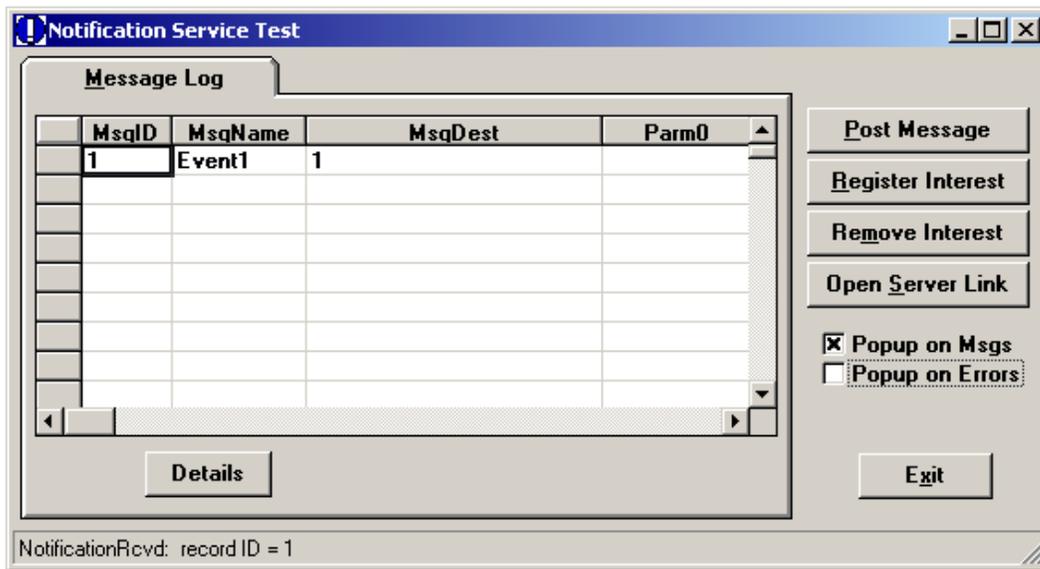
3.2.4 PC Notification Test Client

3.2.4.1 PC Test Client – NOTITEST.EXE

The PC Notification Test Client, `NOTITEST.EXE`, is included in the FEMIS installation and can be used to test notification functions and diagnose notification problems. This program enables a user to manually post notification events, monitor events generated by other applications, and force notification errors for testing purposes. See the Section 3.2.4.4, PC Test Client Functions, for more information.

At startup, `NOTITEST` automatically establishes a notification client link and registers an interest in the `Event1 : 1` event. It also enables notification loopback so it can receive its own events. However, `NOTITEST` starts in stand-alone mode, without connecting to a UNIX notification server. Use the `OpenServerLink` function if you wish to open a link to your notification server.

Figure 3.2. Notification Service Test Window



3.2.4.2 PC Test Client Configuration

The PC Test Client has no configuration options or other means to customize its default behavior. However, the test functions (below) can be used to change client behavior at runtime.

3.2.4.3 PC Test Client Command-line Options

The PC Test Client has no command-line options.

3.2.4.4 PC Test Client Functions

The PC Test Client offers a variety of functions for testing the Notification Service. These functions are accessible through command buttons on the test client user interface window.

Open Server Link

The `Open Server Link` function opens a link between the PC Notification Service and a named notification server. The user is prompted for the server name and port number. When the user clicks the `OK` button, the Notification Service closes the previous server link (if any) and sends a connection request to the new notification server.

If the server is available, a connection is established and this server becomes the notification server for this PC. If the server is not available, the Notification Service will ask whether you wish to retry the connection. If you select `Yes`, the Notification Service will treat the problem as a lost connection and go into restoration mode. Otherwise, the Notification Service will go into stand-alone mode and operate without a server connection.

This function is enabled at all times and is useful for testing server connections and simulating lost connections.

Register Interest

The `Register Interest` function enables the test client to register an interest in one or more notification events. The user is prompted for a message name and message destination that uniquely identify a notification event. When the user clicks the `OK` button, the test client registers an interest in the specified event and begins to log all notifications for that event.

Note: To monitor all messages, enter `ALL` for the message name and message destination.

This function is very useful for troubleshooting notification problems because it allows the user to monitor notification events posted by other applications. For instance, if an application is not responding to a specific sequence of notification events, the test client program can register an interest in those events and verify that the events are being sent in the correct order.

This function is enabled only when the test client has a valid client link.

Remove Interest

The `Remove Interest` function enables the test client to remove an interest in one or more notification events. The user is prompted for a message name and message destination that uniquely identify a notification event. When the user clicks the `OK` button, the test client removes its interest in the specified event and is no longer notified about that event.

This function is enabled only when the test client has a valid client link.

Post Event

The `Post Event` enables the test client to post a notification event and simulate events posted by other applications. The user is prompted for the event name, exercise number, and three event parameters, along with control flags that determine how the event will be processed. When the user clicks the `OK` button, the test client sends this event to the Notification Service for distribution to other local and remote clients.

This function is very useful for troubleshooting notification problems because it allows a user to simulate notification events posted by other applications. For instance, the test client can post a specific sequence of notification events and verify that other applications respond correctly to that sequence.

This function is enabled at all times.

Popup On Event

The `Popup On Event` option is used to alert the user each time the test client receives an event notification. This allows the test client to function as an event monitor by displaying a popup message box whenever an event is received. This function can also test the Notification Service queuing functions by introducing a user-controlled delay into the event processing system.

Popup On Errors

The `Popup On Errors` option facilitates error-handling tests by displaying a popup message each time an invalid notification message is received.

3.2.4.5 PC Test Client Diagnostics

The PC Test Client does not include any diagnostic functions.

3.2.5 Notification Server Troubleshooting

The notification server is very stable; however, this program runs in a network environment and, thus, is prone to any and all failures that can occur in network computing and distributed data management systems.

3.2.5.1 Check Notification Server Active

To check if the notification server is active, log in to the UNIX server and issue the following command:

```
%/usr | ucb | ps axw | grep femis_event
```

If the notification server is active, you will get a reply such as:

```
17739 pe S 0:00 femis_event -c server1 -e femis_event.e.log.941219.1140  
  
1073 pe S 0:00 grep femis_event
```

If the notification server is not active, only the second line above will be displayed. The process identification (PID) number of the `femis_event` notification server is the first number shown, e.g., 17739.

3.2.5.2 Check Notification Server Communication

To check the notification server communication, run the UNIX test client either from the server host or from another UNIX system. You should be able to run `fev` and issue notification server instructions. Example:

```
% fev
```

If the notification server is not active, you will get a reply such as the following and then be returned to the command-line processor:

```
fev - test client for femis_event server  
FEMIS_EVENT port is 9020  
connect failed: Connection refused  
%
```

If the notification server is active, you should get a reply such as the following:

```
fev - test client for femis_event server  
FEMIS_EVENT port is 9020
```

After receiving the above reply, you can issue an instruction to the UNIX test client. Example:

```
o
```

This is the test client's command to open a link. Next you should see

```
<<<<<< received OPENLINK-reply: client-id = nnnn
```

where `nnnn` is an open link ID number (could be any positive integer).

If you get such a reply, the notification server is active and communicating. If the notification server is active and communicating, then the problem is probably either in the network or on the PC side.

3.2.5.3 Aborting Notification Server

If you need to abort the notification server during testing or troubleshooting, you must manually log in as the user account from which `femis_event` was started. In FEMIS, the user account is `femis`, or you can log in as `superuser`.

You next need to learn the PID number of the `femis_event` server needing to be killed. There are two ways to learn the PID of a FEMIS server process.

The first is to use the `ps` and `grep` commands. Example:

```
%/usr | ucb | ps axw | grep femis_event
```

If the notification server is active, you will get a reply such as:

```
23473 pe S 0:00 femis_event -c server2 -e femis_event.e.log.941219.1140
1073 pe S 0:00 grep femis_event
```

If the notification server is not active, only the second line above will be displayed. The PID of the `femis_event` notification server is the first number shown, e.g., `23473`.

The second way to learn the PID of `femis_event` is to run the test client and use the `!i spy` command. Example:

```
% fev - # connect to local host
fev - test client for femis_event server
FEMIS_EVENT port is 9020
!i
pid . . . . . 23473
```

From the `!i` reply, the `femis_event` pid is `23473`.

With the PID number, you can abort the notification server. The preferred way is

```
% kill -2 23473
```

Recheck if the server is still active. If the above `kill -2` (the graceful exit), did not work, then use

```
% kill -9 23473
```

Using `kill -9` will kill the notification server, but the state of open connections will be lost and possibly may not be recoverable until some long TCP/IP timeout period elapses.

A script file, such as the following, may be used

```
foreach killnum ( -2 -9 )

ps ef >! ..PS..

set serverpid = ( `fgrep femis_event ..PS.. | awk `{print $2}' ` )
foreach pid ( $serverpid )
echo kill $killnum $pid
kill $killnum $pid
end

end
```

3.2.5.4 Fixing Notification Port

When running a FEMIS client application (such as a Visual Basic application), the application first uses the `FEMIS.INI` file in the Windows directory to get the notification server's name and port number. If either the name or port number is incorrect, you will get an error 10054. You could fix the file to avoid this error occurring in the future; but it is not necessary because the Visual Basic application then lets you login to an EOC and gets a new notification server name and port number from the FEMIS database. If either the new name or port number is incorrect, you will get an error 10054. You **must** then correct the EOC table by changing the values for either the `EOC_SERVER_NAME` or the `EOC_NOTIFY_PORT` fields.

3.2.5.5 PC WinSock Errors

The following list includes the errors encountered during development and testing of the notification server software. A complete list of WinSock and UNIX errors can be found in *Windows Sockets, Version 1.1* documentation.

PC WinSock Error 10022

This error is an internal Windows Sockets error which is caused when a Windows application crashes/terminates without properly closing down. In doing so, the Windows application has wasted and lost critical dynamic memory. Error 10022, which means invalid argument, is reported by mistake. The real problem is Windows running out of a critical resource. Shut down other Windows applications and reboot the PC.

PC WinSock Error 10024

This error is an internal Windows Sockets error which is caused when a Windows application crashes/terminates without properly closing down. In doing so, the Windows application has wasted and lost critical dynamic memory. Error 10024, which means too many files open, is reported by mistake. The real problem is Windows running out of a critical resource. Shut down other Windows applications and reboot the PC.

PC WinSock Error 10038

This error is an internal Windows Sockets error that is caused by a software error, most likely manifested from Windows running out of a critical resource. In reaching this error, an application

has tried to reuse an I/O channel that was previously connected to a network socket but has since been closed. Restart the affected applications. If this does not fix the problem, reboot the PC.

PC WinSock Error 10050

This error means the network is down; there is no network communication with the server host to which this PC is trying to connect. Report the error to the System Administrator and wait for a diagnosis. After all hardware and communication bugs have been fixed, restart the affected applications. If this does not fix the problem, reboot the PC.

PC WinSock Error 10053

This error means that connection to the server was aborted and may be because the server was terminated, either intentionally or by a failure. This error can also mean that connection was never established because the server is not currently active. Check if the notification server, `femis_event` is currently active on the UNIX server. If not, restart it using scripts described in Section 3.1.1.5, Daemon Server Startup. The UNIX test client can be used to check for server health, see Section 3.1.4, Notification Server Utilities.

PC WinSock Error 10054

This error means that the notification server is not active. Check if the notification server, `femis_event` is currently active on the UNIX server. If not, restart it using scripts described in Section 3.1.1.5, Daemon Server Startup. The notification subsystem UNIX test client can be used to check on server health, see Section 3.1.4, Notification Server Utilities.

This error can also mean that the client software on the PC does not have the correct service port number or server. The default port for the notification server is 9020. Client software must use this same service port. If the port number is determined to be incorrect, fix it and restart the client software applications. Reboot the PC if necessary.

3.3 Starting/Stopping Notification Service

When the server is rebooted or shutdown, it runs the `/etc/init.d/femis` script, which start or stops the Notification Service using the following scripts in the `/home/femis/bin` directory.

3.3.1 Starting Notification Service

The `/home/femis/bin/start_notify` script uses the EOC List File (`./etc/eoclist.dat`) to determine how to start the Notification Service. The file tells how many Notification Service processes to start, which ports to use, and which other Notification Services to communicate with. You can run the following script.

```
% startnotify
```

If the Notification Service(s) is already running, you cannot start new ones.

To start the Notification Service(s) with logging turned on, you can run the following script:

```
% startnotify -log
```

3.3.2 Stopping Notification Service

The `/home/femis/bin/stopnotify` script stops the Notification Service(s) by finding all processes running the `femis_event` program and then kills them using `kill -2`. You can run the following script.

```
% stopnotify
```

3.4 Data Transfer Notification

Data Transfer Notifications are used to acknowledge the receipt of data. Chemical Accident or Incident (CAI) notifications, Work Plans, D2PC cases, Threat Areas, Risk Areas and Protective Action Recommendations (PARs) are broadcast from onpost to offpost. The Data Transfer Notification sends data receipt acknowledgements from the offpost EOCs back to the onpost EOC. When the data has been sent, a Data Acknowledgement Notification window will be displayed on the sending PC. This window will update itself by looking for notifications sent by the receiving server. When the notification is received, a Data Acknowledgement record will also be written to the Shared Journal for historical reference. The FEMIS Notification Service will need to be started in order to run Data Transfer Notification.

3.4.1 Data Acknowledgement Notification Window

When data is broadcast offpost or when a CAI is declared through FEMIS, the Data Acknowledgement Notification window will be displayed. As each server receives the data, a notification will be sent back to the originating server, and the window will be updated with the date and time the information was received. If the offpost server does not receive the data within approximately 6 minutes, the window will be updated with a Timed Out message.

Note: This window will never display Data Acknowledgements from EMIS. Use the Data Acknowledgement Monitoring window to receive EMIS Data Acknowledgements.

3.4.2 Data Acknowledgement Monitoring Window

The Data Acknowledgement Monitoring window can be accessed from the `utility` menu on the Workbench. It will display all the received Data Acknowledgements as they arrive. As each server receives the data, a notification will be sent back to the originating server, and the window will be updated with the date and time the information was received. If the offpost server does not receive the data within approximately 6 minutes, the window will be updated with a Timed Out message.

Note: This window will display Data Acknowledgements for data received from EMIS.