Help Desk Management System
System Administrator's Guide

DRAFT - VERSION 3.0

The purpose of this section of the Administrator's Guide is to familiarize the reader with the responsibilities of the administrator of the application. The sub-sections are listed in no particular order, but they have descriptive titles so that you can find information quickly and easily. Even though it is probably very boring reading, it would be a good idea to read this over a couple of times so you will be aware of what type of problems can occur. [Note that this section is not written at a basic level as were the other sections. It is assumed that whomever is doing the administration of the database knows about things such as processes, `.logins`, file permissions, etc.]

How Users Start the Application

Although the administrator could decide to have users directly execute the ACCELL/Manager themselves, passing to that program the name of the application, this is generally too much to ask of the typical user. At Delmarva Power, there are three departments who use the application, most whom are UNIX novices. An executable script for the C Shell, called `hdms`, was installed in `/usr/local/bin`. To make sure that the environment is set up correctly, this shell script sets all necessary environment variables before starting the application. The script is shown below:

```
#
#  This script starts up the HelpDesk Management System
#
#  Set environmental variables
setenv  DBPATH   /usr/db/ACL4/db
setenv  UNIFY    /usr/db/ACL4/lib
setenv  TERMCAP  /usr/db/ACL4/lib/termcap
setenv  AMGRINIT 'heap_size=65536 init_mem_size=92160'
set path = (/usr/db/ACL4/bin $home/bin $path .)
#  If SPOOLER variable is not previously defined, use default
#  setenv  SPOOLER  'lp -s -d ncclaser -o nobanner'
#  Run application
cd /usr/db/ACL4/aclenv/netops
AMGR netops
clear
#  When reconfiguring
#       (1)  Uncomment the echo line below
#  AND
#       (2)  Comment out the cd and AMGR lines above
#  echo HDMS will be down for about half an hour.  Sorry.
```

All but one of the departments at Delmarva have no use for the UNIX system for anything other than HDMS. As a result, these users had their `.login` make all necessary settings to their environment and then automatically call a menu program, written as a C Shell script. This menu program would allow the users to start HDMS by calling the above-mentioned shell script, change their password, read their mail, take an ACCELL tutorial, or `logout`. The script is shown below:

```
#
set elabel=TOP
TOP:
clear
echo "\n\n                          UNIX SystemPro Application Menu\n\n"
echo "     1.  HDMS    Execute Help Desk Management System\n"
echo "     2.  Tutorial       Execute Accell Tutorial (CTRL-C to exit)\n"
echo "     3.  Passwd  Change your password (Can only be done every
                          14 days)\n"
echo "     4.  Exit    Logout of UNIX\n"
echo "\n             If you have problems or questions regarding:\n"
echo "                 UNIX or this menu contact Blaine Boyles x5224\n"
echo "                 HDMS or Accell contact John Stewart x5319\n"
echo "Enter option #:  \c"
set opt=`line`

switch ("$opt")
case 1:
        hdms
        breaksw
case 2:
        if ( -e ~/accelltut ) then
          cd ~/accelltut
        else
          mkdir ~/accelltut
          cd ~/accelltut
        endif
        accell
        breaksw
case 3:
        clear
        passwd
        breaksw
case 4:
        set elabel=STOP
        breaksw
default:
        echo "\n\nError:  Undefined selection"
        breaksw
endsw
goto $elabel

STOP:
unset opt elabel
```

The Network Operations department's `.logins` would simply set up their environment and present them with the C Shell prompt. This was done because Network Operations uses UNIX for other applications. When these users want to start HDMS, they simply enter `hdms` at the UNIX shell prompt, and the application starts (assuming that `/usr/local/bin` is in their search path).

Zombie Processes

Because of the way that UNIFY wrote the ACCELL/Manager (AMGR), interesting things happen when a user's login shell dies while the user is still in AMGR; this can happen as a result of a network problem, an unclean break by the user, etc. UNIFY chose to write AMGR so that if its parent dies, it changes its PPID from the PID of the login shell to 1 (init's PID). Because the user does not have a shell, he/she will not be listed as logged in by any of the `who`, `w`, `finger`, or `users` commands. These processes must be manually killed by the Super User with the -9 option; `init` will be running as long as the system is up, so the processes will never die of their own accord. In addition to taking up memory and CPU time, zombie `AMGR` processes present two problems. First of all, if the user had a record on the screen when the process became a zombie, that record will be locked and no one else will be able to look at it until the lock is removed by killing the process. The second problem is that when a particular schema (database design) is being reconfigured, no one can be using any application that references data indexed by that schema. Normally, when the database is brought down to be reconfigured, the `w` command is used to make sure that no one who is logged in is running `AMGR` or any other similar programs. If zombie `AMGR` processes are running, you can be tricked into incorrectly thinking that nobody is using the database. The

program that does the reconfiguration realizes that there are AMGR processes running and doesn't do the reconfiguration, but it is up to you to find those processes (via the ps command) and kill them.

Editor's Note: We need to write a script to do this clean up periodically via cron!

Reconfiguration Process

The administrator will have to reconfigure the database whenever he/she wants a change in the schema (database design) to be reflected in the application. This must be done if a field is added, deleted, or changed in size. When the database is being reconfigured, no users may be using the application. The easiest way to check on this is to execute the w command, however you should see "Zombie Processes" above for an important note. [Note that you either need to be the Super User or have mem authorization in order for w to list information on users other than yourself.] Once you have verified that nobody is using the application, you want to make sure that no users enter the application during your work. The easiest way to do this is to edit the file /usr/local/bin/hdms (or whatever you plan on using at your site) to stop people from entering the application; you might also want it to give the user some time frame for when the application will be back up. Note that you may refer to the example /usr/local/bin/hdms script shown above for how we achieved this. After doing this, check one more time to ensure that nobody is using the application. You may now go into UNIFY and reconfigure. During reconfiguration, the program might ask you several questions. If it asks if it should rebuild indexes such as Hash Tables or B-Trees, you should answer "yes" as long as you are not in a hurry. It does take a little longer, but it makes data access quicker and cleans up the data structure. It may also ask if you want to use diskette/tape as a temporary file. As long as you have an ample amount of space in the file system, you should answer "no". Multiple copies of the data dictionary and application data are kept in temporary files during reconfiguring, and it is possible that these temporary files could fill up the file system. Once you are told that the database has been reconfigured, you will have to go into the ACCELL/Environment and both "Adjust Forms" and "Recompile All" forms and scripts for applications that reference fields/tables in the schema just reconfigured. After this is done and you have tested the application, you may edit /usr/local/bin/hdms to allow people to re-enter the application. Note that you may enter the application from within the ACCELL/Environment without having to edit the startup script.