114

AIX update

April 2005

In this issue

A itc PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs $275.00 in the USA and Canada; £180.00 in the UK; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 2000 issue, are available separately to subscribers for $24.00 (£16.00) each including postage.

AIX Update on-line

Code from *AIX Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/aix; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Restore a volume group with a different configuration

This article explains how to recreate and restore a user-defined volume group (non-rootvg) using savevg and restvg with different definitions or characteristics. For example it explains how to create a savevg with a larger PP size than defined for the volume group, and how to restore a savevg from a volume group that had a smaller PP size than the target disks. This is practically useful if you are moving the volume group to a different system with different characteristics, such as bigger disks. You can even use this procedure to recreate or restore the volume group on the same system when you want to change a volume group definition that is not changeable after the initial creation of the volume group, such as PP size.

Here are some volume group definitions that you can change with this procedure:

- Creating or restoring a volume group with a larger PP size. This is helpful if you are moving the volume group to bigger disks that are not supported by the current PP size, on a new or the same system.
- Creating or restoring a volume group without mirroring, on a new system that does not have the same number of disks.
- Creating and restoring the volume group and shrinking the filesystem sizes at the same time.
- Resizing one or more filesystems when restoring the volume group.
- Changing mount points for some logical volumes when restoring the volume group on another system.

There are many other things that are not included in the above list.

For the rootvg volume group, a similar procedure can be used with the **mksysb** back-up, but we are not going to explain that in this article; it will be left to a future issue. So let us start with this procedure:

1 Create a new */tmp/vgdata/<vgname>/<vgname>.data* file by running the following command:

```
mkvgdata <vgname>
```

2 Edit the file */tmp/vgdata/<vgname>/<vgname>.data*:

```
cd /tmp/vgdata/<vgname>
vi <vgname>.data
```

3 The following examples show various possibilities of what can be changed in the volume group definition. The examples are not the only possibilities; there are other definitions that can be changed as well, but the examples show those that are most common. The changed lines are in italics.

Example 1:

Changing the PP size of a volume group from 8 to 16:

```
vg_data:
    VGNAME=myvg
    PPSIZE=8   => PPSIZE=16
    VG_SOURCE_DISK_LIST=hdisk1
    QUORUM=2
    CONC_CAPABLE=no
    CONC_AUTO=no

lv_data:
    VOLUME_GROUP=
    LV_SOURCE_DISK_LIST= hdisk1
    LV_IDENTIFIER= 00000000067997cd1.1
    LOGICAL_VOLUME= lv100
    VG_STAT= active/complete
    TYPE= jfs
    MAX_LPS= 512
    COPIES= 1
    LPs= 10  => LPs=5
    STALE_PPs= 0
    INTER_POLICY= minimum
    INTRA_POLICY= middle
```

```
MOUNT_POINT= /apps/data
MIRROR_WRITE_CONSISTENCY= on
LV_SEPARATE_PV= yes
PERMISSION= read/write
LV_STATE= opened/syncd
WRITE_VERIFY= off
PP_SIZE= 8  =>  PP_SIZE=16
SCHED_POLICY= parallel
PP= 10  =>  PP=5
BB_POLICY= relocatable
RELOCATABLE= yes
UPPER_BOUND=32
LABEL=/apps
MAPFILE=
LV_MIN_LPS=7  =>  LV_MIN_LPS= 4
```

Note that in this example the PPSIZE in the vg_data stanza has been changed from 8 to 16. Also the PP_SIZE in the lv_data stanza has changed from 8 to 16. The LPs and PP in lv_data were divided by 2 to reduce the number of LPs that the logical volume has. Remember that you have to make the same changes to PP_SIZE, LPs, PP, and LV_MIN_LPS for each lv_data stanza for each logical volume in the volume group.

The division of the LPs, PP, and LV_MIN_LPS is necessary so that the logical volume is created with the same size. Figure 1 is a division table to use as a reference.

Old PP size	*New PP size*	*Divide LPs, PP, and LV_MIN_LPS by*
4	8	2
4	16	4
4	32	8
8	16	2
8	32	4
8	64	8
16	32	2
16	64	4
32	64	2
32	128	4
…		

Figure 1: Reference division table

If the result of the division is a decimal number, eg 4.5, round it up to the nearest whole number. You can also reverse this example if your intention is to make the PP size smaller instead of larger.

Note: AIX 5L provides an option (**-p PPsize**) with the **restvg** command that will override and change the PP size recorded in the */tmp/vgdata/<vgname>/<vgname>.data* file. Therefore you might not need this procedure to change the PP size when you restore and it will change the number of the partition appropriately, depending on the PP size.

Example 2:

Changing the number of copies and removing mirroring:

```
vg_data:
    VGNAME=myvg
    PPSIZE=8
    VG_SOURCE_DISK_LIST=hdisk1
    QUORUM=1  =>  QUORUM=2
    CONC_CAPABLE=no
    CONC_AUTO=no

lv_data:
    VOLUME_GROUP=
    LV_SOURCE_DISK_LIST= hdisk1
    LV_IDENTIFIER= 0000000067997cd1.1
    LOGICAL_VOLUME= lv100
    VG_STAT= active/complete
    TYPE= jfs
    MAX_LPS= 512
    COPIES= 2  =>  COPIES=1
    LPs= 10
    STALE_PPs= 0
    INTER_POLICY= minimum
    INTRA_POLICY= middle
    MOUNT_POINT= /apps/data
    MIRROR_WRITE_CONSISTENCY= on
    LV_SEPARATE_PV= yes
    PERMISSION= read/write
    LV_STATE= opened/syncd
    WRITE_VERIFY= off
    PP_SIZE= 8
    SCHED_POLICY= parallel
    PP= 20  =>  PP=10
```

```
        BB_POLICY= relocatable
        RELOCATABLE= yes
        UPPER_BOUND=32
        LABEL=/apps
        MAPFILE=
        LV_MIN_LPS=7
```

Note that in this example the COPIES in the lv_data stanza have been changed from 2 to 1. Also the PP in lv_data was divided by 2 to reduce the number of PPs because the number of copies was reduced by half. Remember that you have to make the same changes to LPs, COPIES, and PP for each lv_data stanza for each logical volume in the volume group.

Finally, the QUORUM in vg_data stanza has been changed from 1 to 2 in order to enable the quorum on the volume group after removing the mirror.

Example 3:

Shrinking filesystems when restoring:

```
logical_volume_policy:
        SHRINK= no  =>  SHRINK=yes
        EXACT_FIT= no
```

Changing SHRINK in the logical_volume_policy stanza will cause the logical volume to be created with the minimum size possible to accommodate the filesystem. This size is specified by the value of the LV_MIN_LPS field of the lv_data stanza.

Note: AIX 5L provides an option (-s) with the **restvg** command that will override SHRINK recorded in the */tmp/vgdata/<vgname>/<vgname>.data* file. Therefore you might not need this procedure to shrink the size of the filesystem.

Example 4:

Resizing certain filesystems only when restoring:

```
      lv_data:
          VOLUME_GROUP=
          LV_SOURCE_DISK_LIST= hdisk1
```

```
        LV_IDENTIFIER= 00000000067997cd1.1
        LOGICAL_VOLUME= lv100
        VG_STAT= active/complete
        TYPE= jfs
        MAX_LPS= 512
        COPIES= 1
        LPs= 10  =>  LPs= 8
        STALE_PPs= 0
        INTER_POLICY= minimum
        INTRA_POLICY= middle
        MOUNT_POINT= /apps/data
        MIRROR_WRITE_CONSISTENCY= on
        LV_SEPARATE_PV= yes
        PERMISSION= read/write
        LV_STATE= opened/syncd
        WRITE_VERIFY= off
        PP_SIZE= 8
        SCHED_POLICY= parallel
        PP= 10  =>  PP= 8
        BB_POLICY= relocatable
        RELOCATABLE= yes
        UPPER_BOUND=32
        LABEL=/apps
        MAPFILE=
        LV_MIN_LPS= 7

    fs_data:
        FS_NAME= /myfs
        FS_SIZE= 163840  =>  FS_SIZE= 131072
        FS_MIN_SIZE= 114688
        FS_LV= /dev/lv100
        FS_FS= 512
        FS_NBPI= 4096
        FS_COMPRESS= no
```

In this example we resize */myfs* from 80MB to 64MB by changing LPs and PP in the lv_data stanza, and changing FS_SIZE in the fs_data stanza for the */myfs* filesystem. Note that the new values are greater than or equal to LV_MIN_LPS and FS_MIN_SIZE. Remember that you have to make the same changes for each lv_data and fs_data stanzas for each logical volume and filesystem in the volume group that you need to resize.

4 Run **savevg** from the command line to use your edited <*vgname.data*> file:

```
savevg -f /dev/rmt# <vgname>
```

where *rmt#* is your tape drive.

Do not run **savevg** from smit. This will update the *<vgname>.data* file, destroying the changes you made. Also, do not run **savevg** with the **-i** flag, because it will do the same thing.

5 Run the **restvg** command to recreate and restore the volume group on the target system you want:

```
restvg -f /dev/rmt# hdisk# hdisk# . . .
```

where *rmt#* is your tape drive and *hdisk#* is the list of disks the VG will be restored to.

If another **savevg** cannot be run, the following procedure can be used to restore a **savevg** to disks with the required changes to the volume group definition:

1 On the target system, place the **savevg** tape in the tape drive.

2 Remove the */tmp/vgdata* directory:

```
rm -rf /tmp/vgdata
```

3 Restore the */tmp/vgdata* directory from the **savevg** tape:

```
cd /
restore -xqvdf /dev/rmt# ./tmp/vgdata
```

4 Follow the examples in step 3 of the preceding section to edit the */tmp/vgdata/<vgname>/<vgname>.data* file and make the necessary changes. Once the changes have been made, continue with step 5 below.

5 Create a back-up by name of */tmp/vgdata* to run **restvg** against:

```
cd /
find ./tmp/vgdata -print | backup -iqvf /tmp/vg.back
```

This should back up the following files:

– */tmp/vgdata/*

– */tmp/vgdata/vgdata.files*

- *./tmp/vgdata/<vgname>*
- *./tmp/vgdata/<vgname>/filesystems*
- *./tmp/vgdata/<vgname>/<vgname>.data*.

6 Verify that the files were backed up, and the file can be read:

```
restore -Tqvf /tmp/vg.back
```

7 Run the **restvg** command to recreate the volume group on the hdisks you want:

```
restvg -f /tmp/vg.back hdisk# hdisk# . . .
```

8 Verify that the volume group and filesystems were re-created, and verify that the filesystems are mounted:

```
lsvg -l <vgname>
mount
```

9 If all looks good, restore the files from the tape:

```
cd /
restore -xqvdf /dev/rmt#
```

where *rmt#* is your tape drive.

Basim Chafik
Senior Systems Analyst
IBM Certified Advanced Technical Expert (CATE)
Plexus (Division of BancTec) (Canada)

Clean up unwanted e-mails from your Unix account inbox

Are your cron jobs filling up your Unix account inbox? If so, you may want to consider setting up your own cron job to delete old e-mails from your Unix account.

The following script can be set up to run periodically through

cron and it will delete old e-mails from your Unix inbox and save your most recent ones.

It can also be used to delete e-mails based on sender or subject line.

I believe the script has been written so that it is easy to understand and modify, if you need to. However, no modifications are necessary if you just want to delete e-mails from root's inbox that are older than 20 days.

CODE

```
#!/usr/bin/perl
##########################################
#####  Purge Unwanted E-mails       #####
#####  Zeida Heavener               #####
#####  January 05, 2005             #####
##########################################

&zopen;
&zprocess;
&zclose;
exit 0;

#############################################################
sub purge {
$pur=0;

# ENTER PURGE CRITERIA HERE by uncommenting lines as indicated
# example of purge based on age
#if ($age > 30) { $pur = 1; }    # purge message if older than 30 days
if ($age > 20) { $pur = 1; }     # purge message if older than 20 days

# example of purge based on subject
#if ($sub_line eq "Subject: abcd") { $pur = 1; }

# example of purge based on beginning of subject field
#$_=$sub_line;
#if (/^Subject: ab/) { $pur = 1; }

# example of purge based on sender
#if ($from eq "zeida") { $pur = 1; }

}
#############################################################
sub zopen {
```

```
# SPECIFY WHOSE INBOX HERE
$infil1 = "/var/spool/mail/zeida";     # for zeida's inbox
$infil1 = "/var/spool/mail/root";      # for root's inbox

# workfiles are:
$wkfil2 = "/tmp/work1.mail";
$otfil3 = "/tmp/mail.out2";

open (FILE1, "$infil1") or die "cannot open input file";
open (FILE2, ">$wkfil2") or die "cannot open output file";

}
############################################################
sub getmess {

while (<FILE1>) {
$linein=$_;
if (/^From \S* [Mon |Tue |Wed |Thu |Fri |Sat |Sun ][Jan |Feb |Mar |Apr
|May |Jun
 |Jul |Aug |Sep |Oct |Nov |Dec ]/) {
&header;
}               # if From
else { print FILE2 "$linein"; }
}               # while

&putmess;

}
############################################################
sub header {
if ($first_time < 1) {    # not first time
&putmess;
}               # if first-time
else {
$first_time=0;
#print FILE2 "$linein";
}               # else
&getdetail;
&purge;
}
############################################################
sub putmess {
close (FILE2);
if ($pur == 0) {
system("cat $wkfil2 >> $otfil3") ;
}
system("cat /dev/null > $wkfil2") ;
open (FILE2, ">$wkfil2") or die "cannot open output file";
}
```

```
############################################################
sub getmon {
# verify month is valid and translate to numeric
$mon_num=Ø;
if ($mon eq "Jan") { $mon_num = 1; }
if ($mon eq "Feb") { $mon_num = 2; }
if ($mon eq "Mar") { $mon_num = 3; }
if ($mon eq "Apr") { $mon_num = 4; }
if ($mon eq "May") { $mon_num = 5; }
if ($mon eq "Jun") { $mon_num = 6; }
if ($mon eq "Jul") { $mon_num = 7; }
if ($mon eq "Aug") { $mon_num = 8; }
if ($mon eq "Sep") { $mon_num = 9; }
if ($mon eq "Oct") { $mon_num = 1Ø; }
if ($mon eq "Nov") { $mon_num = 11; }
if ($mon eq "Dec") { $mon_num = 12; }
if ($mon_num == Ø) { die "invalid month" ; }
}
############################################################
sub getdetail {
@det1 = split;
$from = $det1[1];
$week = $det1[2];
$mon  = $det1[3];
$day  = $det1[4];
$year = $det1[6];
&getmon;
foreach (1, 2, 3, 4, 5, 6) {
print FILE2 "$linein";
$linein= <FILE1> or die "error in input file format\n" ;
}
$sub_line=$linein;
chomp $sub_line;
&getage;
print FILE2 "$linein";
}
############################################################
sub getage {
&today;         # $yy  $jul
&getjul;        # $fyy $fjul
#print $fyy, " ", $fjul, "  ", $yy, "  ", $jul, "\n";
{if (($yy == $fyy) && ($jul >= $fjul)) { $age = $jul-$fjul; last;}
if ($yy > $fyy) { $age = ((($yy - $fyy)* 365)+ $jul)-$fjul; last;}
die "date is wrong";
}
#print $age, "\n";
}
############################################################
sub today {
($SS,$MM,$HH,$dd,$mm,$yy,$wk,$jul,$dst) = localtime(time);
```

```
$yy = $yy + 1900;
$mm = $mm + 1;
$jul = $jul + 1;
#print $yy, "\n", $jul, "\n", $mm, "\n", $dd, "\n";
}
##############################################################
sub getjul {
# input is      $mon_num  $day  $year
# output is     $fyy $fjul
$fyy = $year;
$fjul = $day;
if ($mon_num > 1) {$fjul = $fjul+31;}
if ($mon_num > 2) {$fjul = $fjul+28;}
if ($mon_num > 3) {$fjul = $fjul+31;}
if ($mon_num > 4) {$fjul = $fjul+30;}
if ($mon_num > 5) {$fjul = $fjul+31;}
if ($mon_num > 6) {$fjul = $fjul+30;}
if ($mon_num > 7) {$fjul = $fjul+31;}
if ($mon_num > 8) {$fjul = $fjul+31;}
if ($mon_num > 9) {$fjul = $fjul+30;}
if ($mon_num > 10) {$fjul = $fjul+31;}
if ($mon_num > 11) {$fjul = $fjul+30;}
if ($mon_num > 2) {
$leap=$fyy%4;
if ($leap == 0) {$fjul = $fjul+1}
}
#print $mon_num, "  ", $fjul, "\n";
}
##############################################################
sub zprocess {

$first_time=1;
system("cat /dev/null > $otfil3");
system("cat /dev/null > $wkfil2");
&getmess;

}
##############################################################
sub zclose {

close (FILE1);
close (FILE2);

system("cp $otfil3 $infil1");
}
##############################################################
```

I/O Wait

In his paper, 'Demystifying I/O Wait', Harold Lee [1] explained where the I/O Wait metric comes from, how it's generated, and what it can mean. Still, a common conclusion other than 'it depends' remains elusive. This article will point out some common reasons why I/O Wait might be reading high and what can be done about these readings. Without reiterating all of the details from Lee's paper, a brief definition of I/O Wait is 'the percentage of CPU time spent waiting on disk I/O'. Less briefly, I/O Wait (iowait) is a special form of the 'idle' metric, which is used when a processor is waiting for an instruction on a process that is sitting in the blocked queue. In other words, if a processor has to wait for a process to get information from disk (or some other I/O bus), I/O Wait is incremented instead of 'idle'. I/O wait is observed in the rightmost *wa* column in vmstat output, the *% iowait* column in iostat, the *%wio* column in the **sar -P**, and the ASCII bar graph titled 'wait in topas'. For sysadmins who haven't spent a whole lot of time designing enterprise systems or redesigning configurations for existing systems, it's a good idea to consider I/O Wait among the many metrics before spending money on new hardware.

So where does I/O Wait come from? I/O Wait can originate from many different conditions, depending on your application and hardware. The first condition we'll discuss doesn't actually have to do with anything more than interpretation and measurement, but because it's very common, it's worth mentioning. Often, when sysadmins look at iostat or vmstat, they look at the cumulative calculations from simply running these commands. This cumulative output can report high or even negative (when the system has been up long enough that the signed data types in the code for those commands eventually overflow as the counter gets larger) I/O Wait. In such cases, accurate I/O Wait statistics can be gathered using the duration and iteration operators for vmstat and iostat.

Example 1: on a four-processor box with a large amount of I/O Wait, a single cumulative reading from iostat will appear as:

```
root@doggone
/# iostat | grep -p tty

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
      -204.5    -34.2             -21.3     -4.2      155.2     -29.8
```

Example 2: on the same box, a single reading from vmstat:

```
root@doggone
/# vmstat
kthr     memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b    avm   fre   re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  1 542760  4543   0   0   0   0    3   0  96 2031 763 10  2 74 14
```

I/O Waits for current processes easily oscillate within the time it takes you to execute the command and the time it takes to write to STDOUT.

Example 3: iostat on the same box using duration and iteration operators yields much more accurate output and reveals I/O Wait higher than we thought using the single run of iostat:

```
root@doggone
/# iostat 1 5 | grep -p tty
tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
      -204.6    -34.2             -21.3     -4.2      155.3     -29.8

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
        89.8    162.6              11.0      4.0       57.9      27.2

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
       416.0    162.0              14.2      3.0       62.5      20.2

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
        28.9    161.6                8.5      3.0       65.6      22.9

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
       422.0    162.0                9.2      5.2       66.0      19.5
```

Example 4: vmstat using duration and iteration operators:

```
root@doggone
/# vmstat 1 5
kthr     memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
```

```
 r  b   avm   fre   re  pi  po  fr   sr  cy  in   sy     cs us sy id wa
 1  1 534404  6618   0   0   0   0    3   0  96 2031   763 10  2 74 14
 2  0 534545  6474   0   1   0   0    0   0 949 8175  1246 33  4 39 24
 2  0 534545  6468   0   0   0   0    0   0 854 12642 1358 44  6 28 22
 1  0 534985  6022   0   1   0   0    0   0 926 11120 1422 46  7 23 24
 2  0 534987  6016   0   0   0   0    0   0 836 20330 1307 59  5 23 13
```

Example 5: even over five iterations, where duration is longer, I/O Wait appears higher:

```
root@doggone
/# vmstat 2 5
kthr     memory             page              faults        cpu
----- ------------ ------------------------ ------------ -----------
 r  b   avm   fre   re  pi  po  fr   sr  cy  in   sy   cs us sy id wa
 1  1 534981  6007   0   0   0   0    3   0  96 2031  763 10  2 74 14
 2  0 535320  5661   0   0   0   0    0   0 872 6638 1215 32  6 37 26
 1  0 535276  5695   0   0   0   0    0   0 845 8130 1352 32 10 37 20
 1  0 535418  5544   0   0   0   0    0   0 857 5033 1256 17  4 55 25
 1  0 535554  5400   0   0   0   0    0   0 844 5581 1281 18  3 49 30
```

The slight modification of measurement technique can alter our results enough to change our management or planning because I/O Waits of 30% (as opposed to 14%) would be more likely to warrant expenditure for better hardware (where warranted; we'll discuss this later). However, this is just the first step.

Another, more obvious, condition that produces high I/O Waits is a bottleneck in your I/O subsystem. Often, I/O Wait is our only indicator of such problems. These could be slow or failing disks or a competent disk array in which logical volumes span multiple physical volumes. Several or disparate disks means that the reading of data from the disks is fragmented. This latency will show up in the form of elevated I/O Wait. In these cases, the solution can be simple, but painfully expensive. And that's a question you'd have to answer in direct relation to your environment and your applications, but inevitably it's a choice that an enterprise has to make. If you're worried about justifying the expense, the I/O Wait metric can be very helpful.

When looking at I/O Wait on systems with both high I/O and high computational loads, it's important to consider the other

numbers in iostat and the blocked queue because a truly computationally busy system can mask I/O problems. In these cases, use the **sar** command for a closer inspection of I/O Wait. Remember that I/O Wait is generally a global statistic and doesn't usually report for any specific processor, but with **sar** you can see such individual measurements. In Example 6, we can see the I/O Wait (wio) by processor and also the average.

Example 6: observing I/O Wait with **sar -P ALL**:

```
root@doggone
/# sar -P ALL

AIX doggone 1 5 000054321C00    12/20/04

16:00:04 cpu    %usr    %sys    %wio   %idle
16:00:04  0       11      15      33      41
          1       15       4      56      26
          2       48       7       4      41
          3       30      52      11       7
          -       24      20      26      30
17:00:04  0       72      14      10       3
          1       64      11      18       7
          2       46      50       4       0
          3       50      32      14       4
          -       59      26      12       3
18:00:02  0       29      54       4      14
          1       79       3       7      10
          2       36      25      39       0
          3       57       7      14      21
          -       50      22      16      12
19:00:05  0       35      23      35       8
          1       30      37      33       0
          2       36      50       7       7
          3       44       7      26      22
          -       47      19      23      11
```

With these measurements, we can determine whether the I/O Wait is increasing independently of the multiple CPUs or whether it's high for all of them. Using multiple processor systems where an application has more I/O threads than CPU threads could tend to increase I/O Wait because all the processors will get charged I/O Wait if there is a process waiting for a disk. For this reason, during your planning process, you should consider using multiple processor systems

only where your CPU load requires it. While most new systems might swallow an application whole after a hardware upgrade, it is a temporary solution with added expense.

These are straightforward examples of one condition or another, but what if you have a high CPU load and a lot of I/O? With a large percentage of enterprise servers running some kind of database, this is a common condition. In AIX, the answer to keeping I/O Wait respectable is easy: install and turn on asynchronous I/O. Asynchronous I/O operations will run in the background and won't block user applications processing. Performance is improved because I/O operations and application processing can run simultaneously.

Example 7: in a system with a lot of I/O that also has a good bit of CPU utilization, AIO keeps I/O Wait to a minimum:

```
root@daggum
/# iostat 1 5 | grep -p tty
tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
        4.2    1202.4              10.6      1.9       87.4       0.1

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
        0.0     163.0                5.2      0.5       94.2       0.0

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
        0.0     162.0                2.8      0.2       97.0       0.0

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
        0.0     162.0                4.8      0.5       94.8       0.0

tty:    tin      tout   avg-cpu:  % user    % sys     % idle    % iowait
        0.0     161.6                2.5      0.2       97.3       0.0
```

Ultimately, the best way to reduce I/O Wait is to design your systems intelligently and keep the technology of system components comparable. Pairing a p650 with Model 10 SSA will reduce costs, but at the price of poor performance. Typically, I/O Wait is observed after a system is put in place, or when looking at new hardware. As a result, I/O Wait is a reflexive metric, ie measuring I/O Wait is an exercise in tailoring a system, and you do so to keep I/O Wait low. This means that the measurement of I/O Wait is just the beginning

of the line in monitoring system performance, but it's a pretty good place to start.

RESOURCES

1 *Demystifying I/O Wait Redbook* – Harold Lee.

2 *AIX 5L Performance Tools Handbook Redbook* – Budi Darmawan, Charles Kamers, Hennie Pienaar, Janet Shiu.

Matt Frye
AIX Administrator (USA)

Successful business continuity: user and group names and numbers

During a Disaster Recovery (DR) implementation effort, the last thing you want is unexpected hardware and software configuration issues. These tend to consume time, resources, and cause Recovery Time Objectives (RTO) to be missed. In order to ensure business continuity, organizations must design, implement, maintain, and enforce policies, procedures, standards, and guidelines that encompass all aspects of their critical business functions.

A successful DR effort is not only dependent on a well thought-out DR plan, also it must have been derived from an enterprise-wide mentality of business continuity. Furthermore, business continuity must be the starting point in systems design, not the end point. Unfortunately, very few systems are built from the business continuity perspective backwards.

This series of articles will discuss AIX/HACMP systems design from a business continuity perspective, and will provide scripting examples to support the suggested policies, guidelines, standards, and procedures.

Business continuity is the activity performed by an organization to ensure that critical business functions will be available to customers, suppliers, regulators, and other entities that must have access to those functions. These activities include many daily chores such as project management, system back-ups, change control, and Help Desk. Business continuity is not something implemented at the time of a disaster; business continuity consists in those activities performed daily to maintain service, consistency, and recoverability.

The foundation of business continuity is the policies, guidelines, standards, and procedures implemented by an organization. All system design, implementation, support, and maintenance must be based on this foundation in order to have any hope of achieving business continuity, disaster recovery, or in some cases, system support.

This series of articles will define many IT areas that require enterprise-wide policies, guidelines, standards, and procedures to be defined, and will offer recommended solutions for those defined areas. The areas discussed will include:

- Article 1:
  - User names and UID numbers
  - Group names and GID numbers.
- Article 2:
  - Machine names
  - Hostnames
  - Boot adapter and service names
  - Resource group names
  - Aliases.
- Article 3:
  - Volume groups

  - Major numbers
  - Logical volumes
  - JFS log logical volume names
  - Mount points.
- Article 4:
  - MQ Series queue names and aliases
  - Resource group start/stop scripts
  - Error logging
  - Error notification.
- Article 5:
  - Automated documentation
  - Console access
  - Job scheduling
  - Project planning.

It is usually a bad idea to attempt to maintain multiple standards in support of a specific IT area. In many organizations today, separate standards are maintained for stand-alone AIX systems and HACMP clusters. This series of articles will show that this is not necessary, and will describe how to consolidate into a single standard for all AIX systems, including HACMP.

Definition: enterprise-wide unique – refers to a parameter that has one distinct value across any or all platforms throughout the entire enterprise.

USER AND GROUP NAME – UID/GID NUMBERS

On an AIX or Unix system, files are not stored by filename, they are stored by inode number. Each file has an inode and is identified by an inode number, sometimes called an i-number, in the filesystem where it resides (see

www.webopedia.com/TERM/I/inode.html). Inodes provide important information about files, such as user and group ownership, access permissions, and file type. Each file in a Unix system is associated with exactly one user (owner) and one group.

Similar to the filename/inode relationship, the file owner and group membership are designated and stored by the UID/GID numbers. Each user on a Unix system is assigned a UID number and each group a GID number. These numbers are used by the inode to assign ownership and group membership to files.

In business continuity planning, it is important to recognize that UID/GID numbers should be uniquely assigned to users and groups on an enterprise-wide basis.

If these UID/GID numbers are not enterprise-wide unique, there will be security issues as well as conflicts during high availability or disaster recovery failovers. Security issues will also arise when a back-up is restored onto a machine where the UID/GID numbers do not match those stored in the back-up.

The following are recommended policies, standards, guidelines, and procedures.

Policies: user and group names – UID/GID numbers

Each user who is given access to any system shall have an enterprise-wide unique username assigned to them. This username will be implemented on each system to which this user requires access.

Each user who is given access to any system shall have an enterprise-wide unique UID number assigned to them. This UID number will be implemented on all systems requiring a username/UID association.

Each group name implemented on a Unix system will have an enterprise-wide unique GID number assigned to it. This GID

number will be implemented on every system where this group name is implemented.

Guidelines: user and group names – UID/GID numbers

It is important to choose a format for user names that provides enterprise-wide flexibility for use with as many platforms as possible. A user name format that appears to be supported on the widest range of platforms (AIX, Linux, AS/400, MVS, VSE, MS Windows, Unix) is a seven-character username, the first three characters being lower case letters, the last four characters being digits from 0 to 9.

In order to avoid the maintenance and support issues of keeping a database of UID/GID values and their associated user and group names, a reproducible algorithm should be used to calculate the UID/GID values. This algorithm should be executable on any Unix system and provide the same UID/GID number for a given user or group name. Since the UID/GID numbers are reproducible on any Unix platform, the only values that must be maintained are the enterprise-wide unique user and group names.

Standards: user names and UID numbers

In order to facilitate normal maintenance, disaster recovery, and business continuity, it is recommended that each enterprise-wide unique user and group name also be assigned an enterprise-wide unique UID or GID number.

This will cause files whose owners/groups are not defined in the /etc/password file to appear in a long listing with the enterprise-wide unique UID and GID numbers for the owner and group. This condition may indicate which users need to be added to the system, or which files need to be removed from the system. Regardless of the fix, file security will remain intact and uncompromised.

Instead of keeping a database of username/UID and group name/GID associations, it is preferred to use an algorithm to

generate the UID/GID values based on the username/group name. This algorithm must be reproducible across all systems that require a UID/GID number.

A typical algorithm for performing a UID/GID calculation is to use the **sum** command with the **-r** option to generate the Berkeley cksum value. The **-r** option is supported by AIX and appears to be implemented consistently across a wide variety of other Unix platforms as well. An example of using this technique (all commands shown in this series of articles are expressed in Korn shell syntax) is:

```
$ print "abc1234" | sum -r
29247  1
```

A limitation with this technique is that it will calculate only numbers between about 600 and 65,000 for the specified username format of three lower-case letters followed by four digits. So the number of users and groups, enterprise wide, is limited to less than 65,000, and the possibility of duplicates does exist with this algorithm. These limitations may be acceptable for some organizations; however, most will want to expand this concept of UID/GID calculation.

The recommended UID/GID calculation algorithm uses a base 26 numbering system to represent the first three characters of the username, and then converts the calculated base 26 number to a base 10 (decimal) value. An example script implementing this technique is shown below.

```
#!/usr/bin/ksh93
##############################################################################
function usagemsg_mkuid {
  print "
Program: mkuid

This function generates a UID number for a username,
that username must consist of 3 letters followed by 4
digits, or 4 letters followed by 3 digits.  The
username is assumed to be the first argument to the
function.  The username must also be exactly 7
characters long.

Usage: ${1##*/} username
```

```
  Where:
    username = XXX#### - 3 letters followed by 4 digits
        Generates UID between 1,000,000 and 176,759,999
             or
    username = XXXX### - 4 letters followed by 3 digits
        Generates UID between 176,760,000 and 633,735,000

Author: Dana French (dfrench@mtxia.com) Copyright 2004
\"AutoContent\" enabled
"
}
#########################################################################
####
#### Description:
####
#### This function generates a UID number for a username, that
#### username must consist of 3 letters followed by 4 digits,
#### or 4 letters followed by 3 digits.  The username is assumed
#### to be the first argument to the function.
####
#### Assumptions:
####
#### This function assumes the username is constructed with
#### either 3 or 4 contiguous alphabetic characters followed
#### by 4 or 3 contiguous digits.  The total number of
#### characters in the user name must be exactly 7.
####
#### Dependencies:
####
#### The "mkuid" function is dependent on the external
#### function "mkascii" to produce an associative array
#### of ascii decimal values for the lower case letters.
####
#### Products:
####
#### Upon successful completion, this function prints a single
#### decimal value to standard output.
####
#### For usernames consisting of 3 lower case letters
#### followed by 4 digits, the resulting UID values are
#### between 1,000,000 and 176,759,999.
####
#### For usernames consisting of 4 lower case letters
#### followed by 3 digits, the resulting UID values are
#### between 176,760,000 and 633,735,000.
####
#### Configured Usage:
####
#### The "mkuid" function can be called from the command
#### line, script, or another function.
```

```
####
#### Details:
####
#########################################################################
function mkuid
{
  typeset -l LET
  typeset NUMLET=Ø

  [[ "${1}" == "-?" ]] && usagemsg_mkuid "${Ø}" && return 1

####
#### If the total number of characters in the username
#### command line argument is not equal to 7, display an
#### error message, followed by the usage message, and return
#### from the function with an unsuccessful return code.
####

  if (( ${#1} != 7 ))
  then
      print -u 2 "ERROR: Number of characters in username ${1} is not
equal to 7\n"
      usagemsg_mkuid "${Ø}"
      return -1
  fi

####
#### Extract only the alphabetic characters from the
#### username command line argument and count the number
#### of characters extracted.
####

  NUMLET="${1//([!a-zA-Z])/}"
  NUMLET="${#NUMLET}"

####
#### If the number of alphabetic characters in the username
#### command line argument is not equal to 3 or 4, display an
#### error message, followed by the usage message, and return
#### from the function with an unsuccessful return code.
####

  if (( NUMLET != 3 )) && (( NUMLET != 4 ))
  then
      print -u 2 "ERROR: Number of letters in username ${1} is not equal
to 3 or 4\n"
      usagemsg_mkuid "${Ø}"
      return -1
  fi
```

```
#### Determine the base 1Ø order of magnitude for the
#### numeric portion of the user name and store this value.

  (( ORDMAG = 1Ø ** ( 7 - NUMLET ) ))

#### Initialize the decimal value of the lower limit for the
#### UID number to one million.  The lowest value allowed for
#### a calculated UID number will be this value.

  typeset LOWLIM=1ØØØØØØ

#### Initialize the decimal value of the UID number using a
#### base 26 calculation.  The lower limit is added to this
#### value to ensure the calculated UID number is greater
#### than the lower limit.

  (( LOWUID = ( 26 ** ( NUMLET - 1 ) * LOWLIM / 1ØØ ) + LOWLIM ))
  (( NUMLET == 3 )) && (( LOWUID = LOWLIM ))

#### Initialize several variables containing values used
#### while iterating through each character of the username.

  typeset BASE=26
  typeset NVALUE=""
  typeset LVALUE=Ø
  typeset BASEORDMAG=Ø
  typeset SUBTOT=Ø
  typeset TOT=Ø
  typeset MULT=Ø

####
#### Define an associative array to contain an ascii table of
#### values, the array index is the alphabetic character, the
#### value is the decimal number associated with the
#### character.  Call the external function "mkascii" to
#### create this array.
####

  typeset -A LETTERS
  mkascii LETTERS

#   print " ${LETTERS[@]}"

####
#### Divide the username into individual characters and store
#### each character in an array.  Iterate thru this array one
#### element at a time to calculate the base 26 value of each
#### alphabetic character and the decimal value of each
#### number.
####
```

```
  eval EACHLET="( ${1//(?)/ \1} )"
  for LET in "${EACHLET[@]}"
  do

####
#### If the current interation character is a lower case
#### letter, subtract the decimal value of 97 from the ascii
#### value for this character (97 is the ascii value of the
#### lowercase letter "a").  This will cause the value of
#### "a" to be zero, "b" = 1, "c" = 2, etc.  Assign a
#### variable to contain this decimal value for the letter.
####

    if [[ "_${LET}" = _[a-z] ]]
    then
      LVALUE="$(( ${LETTERS[${LET}]} - 97 ))"

####
#### If the current iteration character is not a lower case
#### letter, then it is a number from Ø - 9.  Set the letter
#### value to zero (indicating the iteration character is not
#### a letter) and append the number to the end of the
#### numeric value.
####

    else
      LVALUE="Ø"
      NVALUE="${NVALUE}${LET}"
    fi

####
#### Calculate a base 26 multiplier for the current iteration
#### of the loop.  Each loop iteration should increase this
#### multiplier by a base 26 order of magnitude.
####

    (( MULT = BASE ** BASEORDMAG ))

####
#### Multiply the decimal letter value by the base 26
#### multiplier and add the product to a running total for
#### each iteration of the loop.
####

    (( SUBTOT = SUBTOT + ( LVALUE * MULT ) ))

####
#### Add a value of 1 to the loop counter, which is used to
#### determine the base 26 order of magnitude.
```

```
####

    (( BASEORDMAG = BASEORDMAG + 1 ))

  done

####
#### Multiply the value calculated for the alphabetic
#### characters by the base 1Ø order of magnitude for the
#### numeric portion of the user name.  Add the numeric
#### portion of the user name and the lower limit value to
#### arrive at a final value for the calculated UID number.
#### Print this value to standard output and return from this
#### function with a successful return code.
####

  (( TOT = ( ( SUBTOT * ORDMAG ) + NVALUE ) + LOWUID ))
  print ${TOT}
  return Ø
}
#############################################################################
function usagemsg_mkascii {
  print "
Program: mkascii

This function accepts an associative array variable name
as the first command line argument, and builds an ASCII
table of characters in that array.  The index of the
associative array is the ASCII character.  The value
contained in each array element is the decimal, hex, or
octal number associated with the ASCII character array
index.

Usage: ${1##*/} ArrayName
  Where:
    ArrayName = Name of a predefined associative array

Author: Dana French (dfrench@mtxia.com) Copyright 2ØØ4
\"AutoContent\" enabled
"
}
#############################################################################
####
#### Description:
####
#### The "mkascii" function returns an ASCII table of
#### characters in an associative array. The index of the
#### associative array is the ASCII character.  The value
#### contained in each array element is the decimal, hex, or
#### octal number associated with the ASCII character array
```

```
#### index.
####
#### The "mkascii" function will accept one or two command
#### line arguments.  The first is the name of a predefined
#### associative array, the optional second command line
#### argument can be specified to designate the type of value
#### (decimal, octal, hexidecimal) to return in the
#### associative array.  Decimal values are returned as
#### default.  If the value of the second command line
#### argument is "8", octal values are returned, if the value
#### of the second command line argument is "16", hexidecimal
#### values are returned.
####
#### Assumptions:
####
#### It is assumed the first command line argument is the
#### name of a predefined associative array.  If the second
#### command line argument is not specified, it is assumed
#### the user is requesting decimal values be returned in the
#### array.
####
#### Dependencies:
####
#### The only valid values for the second command line
#### argument are "8" (octal)  and "16" (hexidecimal).
#### Any other values for the second command line argument
#### will result in an error.
####
#### Products:
####
#### The product of the "mkascii" function is the completion
#### of a predefined associative array indexed by the ASCII
#### character whose value is the decimal, octal, or
#### hexidecimal number that represents the ASCII character.
####
#### Configured Usage:
####
#### The "mkascii" function can be called from the command
#### line, script, or another function.
####
#### Details:
####
###############################################################################
function mkascii
{

####
#### If the first command line argument is a literal "-?",
#### display the usage message and exit the function with an
#### unsuccessful return code.
```

```
####

  [[ "${1}" == "-?" ]] && usagemsg_mkascii "${Ø}" && return 1

####
#### Establish a value variable to contain a decimal value associated
#### with each ASCII character.  The variable containing the
#### decimal value is unevaluated because the evaluation will
#### take place later.
####

  typeset VAL='${CNT}'

####
#### If the second command line argument is "8", change the
#### value variable to contain a statement that will be
#### evaluated later to octal values.
####

  [[ "_${2}" != "_" && "_${2}" =  "_8" ]] && VAL='$( printf "%o"
Øx${i}${j} )'

####
#### If the second command line argument is "16", change the
#### value variable to contain a statement that will be
#### evaluated later to hexidecimal values.
####

  [[ "_${2}" != "_" && "_${2}" = "_16" ]] && VAL='${i}${j}'

####
#### Using the first command line argument as the name of a predefined
#### associative array, create a name reference to that array.
####

  nameref ASCII_TABLE="${1}"

####
#### Initialize a loop counter incremented by one each time
#### through the loop.  This counter is used to represent the
#### decimal value of the ascii character.
####

  typeset CNT=Ø

####
#### Loop through the double digit hexidecimal values for all
#### ASCII characters.
####
```

```
  for i in Ø 1 2 3 4 5 6 7 8 9 A B C D E F
  do
    for j in Ø 1 2 3 4 5 6 7 8 9 A B C D E F
    do

####
#### Evaluate the ASCII character from the double digit
#### hexidecimal value for the current iteration of the
#### loops.  Also evaluate the decimal, octal or hexidecimal
#### value associated with the ASCII character.  Assign
#### the resulting value to the associative array using the
#### "nameref" variable.
####

      eval ASCII_TABLE[\\$(print - $( printf "\\\Ø%o" Øx${i}${j} )
)]=\"${VAL}\" 2>/dev/null

####
#### Increment the loop counter by one (used to represent
#### decimal values of each ASCII character
####

      (( CNT = CNT + 1 ))

####
#### Return to the beginning of the loop to evaluate the next
#### character in the ASCII sequence.
####

    done
  done
}
#########################################################################
####
#### Call the "mkuid" function with all command line arguments

mkuid "${@}"
```

In the *usage message* areas of the above script, the phrase ‘AutoContent Enabled’ appears. This refers to a technique of commenting scripts in such a way that documentation can be generated automatically from the script. This has the added benefit that whenever updates to the script are made, the documentation is automatically updated also. Notice that all comments in the script begin with four hash marks (#) followed by a space, this pattern is used to designate text used as documentation. This automated documentation technique,

referred to here as ‘AutoContent’, will be discussed in a later article.

The mkuid script shown above is composed of shell functions so that, if desired, the script can be separated into individual components and called from a shell function library. Otherwise, it can be executed as-is from the command line.

Standards: group names and GID numbers

Since the group names are unlikely to conform to the same standard selected for the usernames, a different mechanism is required to provide the group name/GID number associations. Software vendors, suppliers, and internal operations will specify group names that must be implemented as-is. For instance the Informix database will require the group name ‘informix’ be implemented on each system running the database.

The recommended UID calculation method that implements a base 26 conversion of the username to a decimal value cannot be used in this instance, because it will generate decimal values greater than that supported by AIX and/or HACMP. Therefore an alternative method is needed for GID calculations. The previously-mentioned Berkeley **sum -r** method will probably suffice for this purpose. The limitation of 65,536 names does not cause a problem in reference to defining group names because it is extremely unlikely that this limitation will ever be reached.

So the recommended algorithm to use for defining the group name/GID number associations is the Berkeley **sum -r** method:

```
$ print "informix" | sum -r
x21883  1
```

Whatever unique UID/GID calculation algorithm is selected, chosen, or written, it is important that it be implemented enterprise wide to eliminate security issues and failover conflicts during high availability or disaster recovery events. This technique also provides added security during normal

maintenance activities by ensuring that files are not accidentally assigned to a user or group that does not own them.

Procedures: user and group names – UID/GID numbers

As part of the standard build for each AIX system, copy the mkuid script to each system. The recommended location is:

```
/usr/local/sh/mkuid
```

When implementing a new user on an AIX system through the SMIT interface, the UID number should first be calculated. An example is shown for a username abc1234:

```
/usr/local/sh/mkuid abc1234
```

The resulting number should be used as the user ID value when adding the user abc1234 to an AIX system through the SMIT interface.

When implementing a new user through a script, the resulting number from the mkuid calculation should be captured in a variable and used as the UID number.

When implementing a new group on an AIX system through the SMIT interface, the GID number should first be calculated using the Berkeley **sum -r** method. An example is shown for the group name informix:

```
$ print "informix" | sum -r | awk '{ print $1 }'
21883
```

The value 21883 should be used as the group ID number when adding the group informix on an AIX system through the SMIT interface.

When implementing a new group through a script, the resulting number from the **sum -r** calculation should be captured in a variable and used as the GID number.

CONCLUSION

Standardizing user names, group names, UID numbers, and GID numbers results in the following:

- Individual login names for each user.
- Centralized user/group management.
- Increased security.
- Improved auditing capabilities.
- Avoidance of user name conflicts.
- Avoidance of group name conflicts.
- Avoidance of UID conflicts.
- Avoidance of GID conflicts.
- Avoidance of security holes during and after DR.

Conflict avoidance is important during a disaster recovery effort because conflicts tend to consume large amounts of time. The last place in which you want to redesign and implement new user/group standards is your disaster recovery site when you are attempting to recover your production systems. For most organizations, their disaster recovery provisions are not an exact duplicate of their production systems. In a disaster recovery implementation, multiple systems may be consolidated onto a single platform. This consolidation will expose conflicts such as user/group names and ID numbers and will require these conflicts to be resolved before the production systems can be recovered.

For those organizations where the disaster recovery provisions are an exact duplicate of their production systems, standardization of user/group names and ID numbers using this technique is also highly desirable for the purpose of simplifying user/group support, maintenance, and security. Again, file security on AIX is enforced by UID/GID numbers, not user/group names; therefore the production and disaster recovery systems must be synchronized on this aspect.

The best way to avoid conflicts during disaster recovery is to implement and enforce policies, guidelines, standards, and procedures to eliminate conflicts as part of your enterprise-wide business continuity planning.

The next article in this series will discuss naming structures for machines, hosts, adapters, resource groups, and aliases for use in a business continuity environment.

Dana French
President
Mt Xia (tel: 615-556-0456) (USA)

Password management system

INTRODUCTION

How do you manage your passwords? We seem to have so many to remember that we tend to start to use very easy to guess passwords or often repeat passwords in cycles. In terms of security, this is not good practice. As system administrators, we tell our users to use passwords that are not easily guessed and not to write them down anywhere, yet we often do exactly what we tell them not to do!

In this article, I will show you how you can easily set up a Web-based application that can be used to store your privileged user passwords in a MySQL database. Once installed and configured, you will have only one password to remember. This makes it easy to keep your passwords difficult to guess and makes it simple for you to change them often.

myPMS is a Web-based password management application, which means it can be accessed from anywhere. It uses simple HTML, PHP, and a MySQL database to store passwords.

This application can be installed on any Web server that can run PHP and MySQL. In this article, I will describe the steps to set this up on an AIX server.

MYPMS FEATURES

myPMS features include:

- Search by hostname or drop-down list.
- Ability to store application and host associated with a password.
- Date inserted is stored.
- Who inserted or updated the password is stored.
- Unlimited password history.
- Password is ‘invisible’ until mouse curser hovers over it.
- Fade to blank page after 60 seconds for additional security.
- Database creation using a simple MySQL batch file.
- HTTPS (SSL) capability if you require.

PREREQUISITES

Here is a list of prerequisite software that needs to be installed on the Web server:

- IBM HTTPD Server or Apache (any version)
- PHP (any version)
- MySQL database (any version).

If you wish to run a secure server running SSL (https), you will need a Web server with the SSL module compiled into IBM HTTPD Server or Apache.

All this software can be found and downloaded from IBM’s AIX Toolbox Download page at http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html.

All the software found on IBM’s AIX Toolbox Download page is in RPM (Redhat Package Manager) format. So, you will need to have the rpm.rte LPP installed before you can install the RPMs. The rpm.rte AIX installp format can be found at ftp:/

/ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/INSTALLP/ppc/rpm.rte.

INSTALLATION ON WEB SERVER

Here are the steps required to prepare the IBM AIX server you have chosen to host the Web-based password application.

Installation of IHS or Apache

In order to run this application, you will need a Web server running Apache or IBM HTTPD Server (IHS). IHS comes standard on the AIX installation CDs. Any version of IHS will work for this project. As another option, you may download Apache from IBM's AIX Toolbox Download page in RPM format.

If you choose to install Apache using the RPM format, simply type the command:

```
# rpm -i apache-1.3.31-1.aix5.1.ppc.rpm
```

Once the LPP or RPM is installed, complete the configuration by modifying the *httpd.conf* file found in the *conf* subdirectory of your installation. By default, the path would be */usr/HTTPServer/conf*.

To verify that your Web server configuration is without errors, type the command:

```
# /usr/HTTPServer/bin/apachectl configtest
```

If no errors are reported, start your Web server with the command:

```
# /usr/HTTPServer/bin/apachectl start
```

We will continue with the configuration of IHS or Apache after the installation and configuration of PHP in the next step.

Installation and configuration of PHP

PHP is a programming language that generates HTML, allowing

you to create dynamic Web sites.

To install PHP, use the rpm command and the **-i** flag as you did in the installation of Apache in the section above.

Once PHP is installed, simply follow the instructions on how to modify your *httpd.conf* file. Basically, the steps required are:

1 Copy the PHP library module in your *libexec* subdirectory of Apache or IHS.

2 Add and load the PHP module in your *httpd.conf* file.

3 Add the new application type so that Apache knows what to do with a file that ends with a *.php* extension.

Here is an example of the lines required from my *httpd.conf* file:

```
LoadModule php4_module        libexec/libphp4.so
AddType application/x-httpd-php .php
```

Now, you are ready to restart your Web server with PHP enabled:

```
# /usr/HTTPServer/bin/apachectl restart
```

Further configuration of Apache or IBM HTTPD server

Now, we must prepare Apache or IHS with the location where we will have our PHP and HTML files stored with the proper security in place.

Here are the basic steps:

1 Create a virtual host in your *httpd.conf* file to your domain name of choice. I will use *mypms.yourdomain.com*.

Example from my *httpd.conf* file:

```
NameVirtualHost *:8¯

<VirtualHost *:8¯>
    DocumentRoot /usr/HTTPServer/htdocs/mypms
    ServerName mypms.yourdomain.com
    ServerAlias mypms
```

```
    ErrorLog logs/mypms-error_log
    CustomLog logs/mypms-access_log common
</VirtualHost>
```

You can modify any of the names or paths for the *VirtualHost* as you wish.

2 Make sure your Web server recognizes the *index.php* start page. Look for a line like this in your *httpd.conf* file:

```
DirectoryIndex index.html index.php index.htm
```

You must now restart Apache or IHS after these changes:

```
# /usr/HTTPServer/bin/apachectl graceful
```

Installation of MySQL

You may download MySQL from IBM's AIX Toolbox Download page in RPM format or compile the source code yourself from http://www.mysql.com.

If you choose to install MySQL using the RPM format, simply type the command:

```
# rpm -i MySQL-client-3.23.58-2.aix5.1.ppc.rpm
# rpm -i MySQL-devel-3.23.58-2.aix5.1.ppc.rpm
# rpm -i MySQL-3.23.58-2.aix5.1.ppc.rpm
```

Do not forget to change the root user password of MySQL:

```
# mysqladmin -h localhost -u root -p password 'newpasswd'
# mysqladmin -h hostname -u root -p password 'newpasswd'
```

Note: at the 'Enter password:' prompt, press *Enter*. Replace *newpasswd* with a hard-to-guess password. Replace *hostname* with the real hostname of your server.

To disable logging into the MySQL database without a user, issue these commands:

```
# mysql -u root -p newpasswd
mysql> use mysql;
mysql> delete from user where user='';
mysql> quit;
#
```

Note: replace *newpasswd* with the real password you set in

the previous step.

Now, start the MySQL database with the command:

```
# /etc/rc.d/init.d/mysql start
```

If you wish to start the database automatically at server start-up, add an entry in */etc/inittab* or create a start-up script.

Remember, before shutting down this server, stop MySQL. To do so, issue the command:

```
# /etc/rc.d/init.d/mysql stop
```

Download myPMS

myPMS is available for download from http://www.lvoware.com/download.html.

You will need minimally to provide your country for statistics purposes only. If you wish to get an e-mail when a new version is released, you may also leave your information.

Installation of myPMS

You are now ready to begin installing myPMS in your DocumenRoot. In my case it is in the directory */usr/HTTPServer/htdocs/mypms*:

1 Use this command to extract the tar file you downloaded from lvoware.com:

```
# cp mypms.tar.gz /usr/HTTPServer/htdocs/
# gzip -dc mypms.tar.gz | tar xf -
```

Note: the extract will create a directory named *mypms*.

2 Go into the directory where you extracted the tar file and you will create the MySQL database and tables needed for the application with the command:

```
# mysql -u root -p < mypms.sql
```

Notes:

- Enter the password when prompted for the MySQL root user.

- For the application login, you do not have to use the MySQL root user account, you can create another one if you wish. This user must have select, insert, update, and delete permissions on the myPMS database tables. Additionally, you can create several users in MySQL for access to this application.

  Tip: if you are not familiar with MySQL administration, get phpMyAdmin. It is available from http://www.phpmyadmin.net.

3 Finally, as a last step, ensure the files in your myPMS DocumentRoot are readable by the user your Web server runs as. If the Web server runs as the user *nobody*, issue these commands:

```
# chown -R nobody /usr/HTTPServer/htdocs/mypms
# chmod -R 755 /usr/HTTPServer/htdocs/mypms
```

TESTING MYPMS

You are now ready to test myPMS. In a Web browser, type the name of the ServerName in the VirtualHost you created for myPMS. In my case, it would be http://mypms.yourdomain.com.

When prompted for a username and password, enter either the MySQL root user account information, or the information of the account you created specifically for this application.

Begin entering your password information!

BACK-UPS

Like any data, make sure you have back-ups in case of a failure. Minimally, I suggest you run a MySQL database dump daily.

A mysqldump script is freely available at http://www.coolcommands.com/index.php?option=com_ccadv&task=

display&id=500.

A MySQL restore script is freely available at http://www.coolcommands.com/index.php?option=com_ccadv&task=display&id=281.

SECURING MYPMS USING SSL

If you wish to secure this Web application using SSL, all you need to do is modify the Apache configuration to load the SSL module, and create your certificate.

This portion is left as an exercise for the user because there are plenty of good documents on how to do this readily available on the Internet.

ADDITIONAL INFORMATION

Additional information on myPMS can be found at http://lvoware.com.

Elvio Prattico
Consultant
PRATTICO Consulting (Canada)

AIX news

LeftHand Networks has announced Release 6.1 of SAN/iQ, its storage operating system that drives the LeftHand SAN.

With this release, the SAN administrator can establish policies that allow a primary storage volume or a volume used to store snapshots to grow as needed until a specified threshold is reached.

This is the first version that supports the AIX operating system.

For further information contact:
URL: www.lefthandnetworks.com/products/dsm.php.

* * *

Neterion, formerly S2io, has announced Xframe E, a new adapter based on the PCI Express bus architecture.

PCI Express is serial, rather than parallel, interconnect technology featuring advantages such as scalable bus bandwidth, lower overhead and lower latency, and improved QoS.

Xframe drivers are available for AIX, as well as Windows, Linux, HP-UX, and Solaris.

For further information contact:
URL: www.neterion.com/news/press/pr050224.html.

* * *

FileTek has announced a port of its StorHouse software components to AIX. StorHouse is a data storage and access management system designed to administer massive amounts of structured and unstructured fixed content, historical data, and their associated back-ups.

StorHouse/SM (storage management component) controls a virtual hierarchy of storage devices to provide a single view of storage to client applications. It is also responsible for automating management tasks such as data migration, back-up, replication, retention, and recovery.

StorHouse/RM (relational database management component) works in conjunction with StorHouse/SM to administer the storage, access, and movement of relational data.

StorHouse/BW is a database extension system for SAP BI data.

For further information contact:
URL: www.filetek.com/press/releases/2005/pr_AIX.htm.

* * *

IBM has created a new D5 software group for AIX 5.2 and 5.3. The ValuePak low-cost version of AIX 5.2 and 5.3, which is missing some features of the full releases, will cost $150 per processor. A regular AIX licence will cost $170 per processor.

This new lower-priced licence is available on the two-way, Power5-based eServer p5 and i5 servers, as well as on the two-way, PowerPC 970-based JS20 blade servers for BladeCenters. Before this announcement, customers had to pay $385 per processor on either a p5 520 or an i5 520.

The new pricing helps AIX to compete more effectively against Linux in terms of software licensing costs.

For further information contact your local IBM representative.