112

AIX update

February 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.

Improving login response with indexed security files

If you have one or more systems with a large number of users and you're typically running with high levels of CPU utilization, you will find that using indexed (hashed) security files will greatly improve login response time and will greatly enhance the performance of applications that use system-level authentication.

The problem is that Unix must read the file serially from start to finish until it finds the required user ID. The */etc/passwd* file can become a performance bottleneck on Unix systems with a large number of users. To mitigate this problem, create indexed security files with the **mkpasswd** command.

Use of **mkpasswd** begins after users are created. Run:

```
# mkpasswd -f
```

Issuing this command will create indexed versions of */etc/passwd*, */etc/security/passwd*, and */etc/security/lastlog* as */etc/passwd.nm.idx*, */etc/passwd.id.idx*, */etc/security/passwd.idx*, and */etc/security/lastlog.idx*.

If you have 96 entries in your */etc/passwd* file, you will receive output like this:

```
/etc/passwd  ->  /etc/passwd.nm.idx
3004-777        Entries processed: 96

/etc/passwd  ->  /etc/passwd.id.idx
3004-777        Entries processed: 96

/etc/security/passwd  ->  /etc/security/passwd.idx
3004-777        Entries processed: 88

/etc/security/lastlog  ->  /etc/security/lastlog.idx
3004-777        Entries processed: 67
```

Once you have built the indexed files, you cannot manually

edit the */etc/passwd*, */etc/security/passwd*, or */etc/security/lastlog* files because the timestamps of the files will not be synchronized with the information contained in the indexed files. This also occurs when the **pwdadm** command is used. In such cases, use the following command to check and rebuild outdated or bad indexes:

```
# mkpasswd -c
```

Had you edited the */etc/passwd* file by hand, and then run the command above, your output would look like this:

```
/etc/passwd  ->  /etc/passwd.nm.idx
3004-777         Entries processed: 96

/etc/passwd  ->  /etc/passwd.id.idx
3004-777         Entries processed: 96
```

Similarly, after any modification to the */etc/security/passwd* file, **mkpasswd** would produce output like this:

```
/etc/security/passwd  ->  /etc/security/passwd.idx
3004-777         Entries processed: 88
```

While these normal sysadmin tasks may create changes that you have to script for, the **passwd**, **mkuser**, **chuser**, and **rmuser** commands, and their smitty counterparts, automatically update the indexed files. Also bear in mind that NIS, LDAP, and DCE user databases are unaffected by **mkpasswd**. However, the fact that these databases are in use may indicate that user authentication at the host level is not appropriate for your application, or at least that somewhere along the line, someone gave the matter a bit of thought.

Matt Frye
AIX Administrator (USA)

To boot, or not to boot: that is the question

I recently visited a site where a huge argument was going on. The AIX staff were heatedly discussing, of all things, reboots.

There were two questions – should they or shouldn't they, and, if they should, how often?

Interestingly, this was quite a big site that had been using AIX for about 10 years. Their original policy had been to reboot every week and now they were rebooting about once a month.

I thought other AIX users might be interested in the arguments that were being put forward by both sides.

The main argument for continuing to reboot regularly was that it was the way they had always done things! The second suggestion was that rebooting helped to clean up the system and remove any hidden processes or fragmented memory. The third reason was that the reboot would identify any hardware or software that had somehow failed, but was hanging on while the system stayed up. This was a failure that was going to happen, and a reboot was a controlled period when it could be identified and dealt with without impacting on the time when users were expecting to be able to work. The fourth argument that was whispered was that weekend reboots generated overtime payments for some and time off for others. The final argument was that regular reboots meant people knew how to reboot. Infrequent reboots meant the person starting the server was probably checking a (possibly out-of-date) printed list of instructions, and could get things wrong – for example, a filesystem doesn't get mounted properly after a reboot.

The anti-reboot people argued that the hardware and software were so robust these days (as opposed to the Windows servers) that there was simply no need to reboot. They also said that if something had gone wrong somewhere (like a system corruption) then the reboot would take a very long time. They also argued that very few user applications need restarting, so these could be left to run. Some of the debaters told horror stories from other sites where the server had been stopped for a while and a disk wouldn't spin up when it was restarted, or power surges on start up that blew components. Others talked of forgotten alarms going off, resulting in various

managers being paged to be told that a server was down. The argument for training and familiarity with the latest procedures when a reboot became necessary was countered by saying that the bulk of the start up was automated anyway. And rebooting NFS servers can be particularly troublesome.

The argument then moved on to maintenance. It was clear, everyone agreed that maintenance was necessary. Some maintenance can be done with the server still running, but where a restart is necessary, the system would be scheduled for a reboot. And this could be many months after the last time.

And that led to a sort of consensus. It was decided to schedule downtime every month in the usual way so that users would expect the system to be down. This time would be used only irregularly when a reboot followed maintenance or the installation of new hardware.

Editor's note: we would be interested in other AIX users views of rebooting and what procedures are in place at their sites.

Independent Consultant (UK)

Help Desk password application

INTRODUCTION

System administrators have enough work to do without having to spend valuable time on tasks like resetting passwords for users. Very often, a Help Desk would be able to do such a task on the spot when the end user calls to open a trouble ticket, except that the system administrators do not want to give privileged access (root) to the Help Desk staff.

In this article, I will show you how you can easily set up a Web-

based application, which can be used by the Help Desk to reset user passwords and too many unsuccessful password attempts on AIX servers.

Of course, this application can be extended to work with any flavour of Unix and user management system (LDAP, NIS, NIS+, etc), but this article will demonstrate how to set up the application in an AIX environment using local password files.

WEB APPLICATION ARCHITECTURE

This Web-based application is written in PHP and HTML. The remote command execution is done using passwordless secure shell (SSH) and the Expect programming language. Security will be enforced using SUDO.

The Web server will be running as the user *nobody*. With the use of SUDO, we will allow the user *nobody* to change user (**su**) to the user *hduser*. *hduser* will be our user who will connect using passwordless SSH to all servers in order to reset the end user's password. Using SUDO, we will limit the user *hduser* to running only the necessary commands and at the same time will provide logging of all commands executed.

Access is secured to this application using HTTPD password authentication, which could optionally be changed to HTTPS if your company requires.

A mechanism also exists to ensure that the Help Desk does not reset generic or privileged account passwords. This feature will be described later. Figure 1 shows the architecture of the Help Desk application.

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 V4.0.6-5 or more recent.

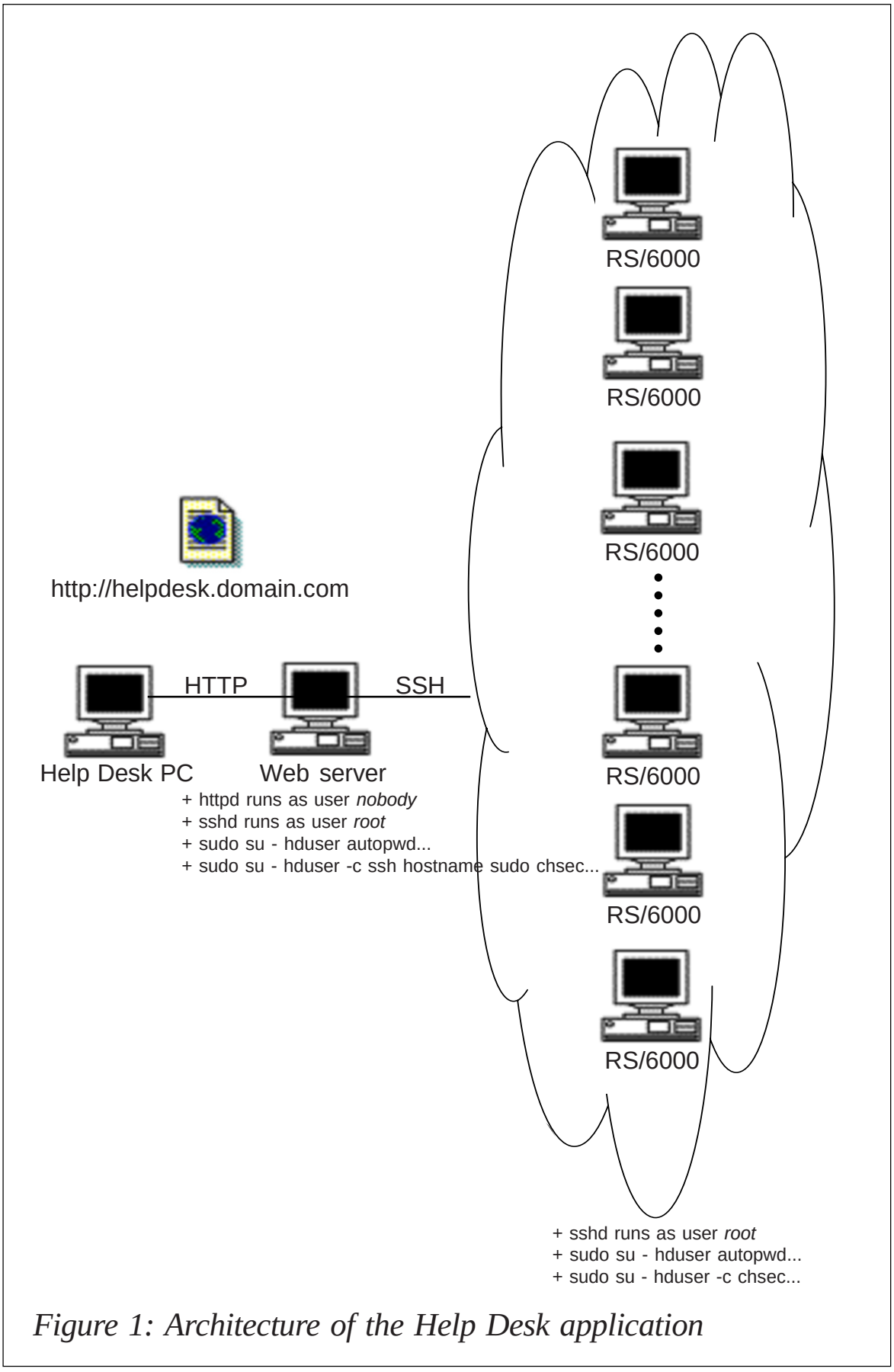

Figure 1: Architecture of the Help Desk application

- Expect V5.34-8 or more recent.
- Tcl V8.3.3-8 or more recent (required by Expect).
- Tk V8.3.3-8 or more recent (required by Expect).
- OpenSSH V3.6p1 or more recent.
- SUDO V1.6.6.0 or more recent.

Here is a list of prerequisite software to be installed on every server that the Help Desk can reset passwords on:

- OpenSSH V3.6p1 or more recent.
- SUDO V1.6.6.0 or more recent.

All this software, with the exception of OpenSSH, 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 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.

OpenSSH can be found on the Bull Freeware site at http://www.bullfreeware.com.

INSTALLATION ON APPLICATION HOSTING SERVER

Here are the steps required to prepare the AIX server you have chosen to host the Help Desk 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 as standard on the IBM 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 above.

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

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
```

More configuration of Apache or IBM HTTPD Server

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

Here are the basic steps:

1 Modify the **ServerName** variable in your *httpd.conf* file to your name of choice. I will use **helpdesk.domain.com**.

This is an example from my *httpd.conf* file:

```
ServerName helpdesk.domain.com
```

2 Protect the DocumentRoot with a username and password scheme.

This is an example from my *httpd.conf* using the default DocumentRoot:

```
<Directory /usr/HTTPServer/htdocs/en_US/>
AuthUserFile /usr/HTTPServer/conf/httpd.passwd
AuthGroupFile /usr/HTTPServer/conf/httpd.group
Satisfy all
AuthType basic
Require group helpdesk
AuthName "Helpdesk Password Utilities"
</Directory>
```

3 Create the appropriate HTTPD password and group files.

To create the HTTPD password file, use the command:

```
# /usr/HTTPServer/bin/htpasswd -c \ /usr/HTTPServer/conf/
httpd.passwd hduser1 hdpwd1
```

Replace the values of **hduser1** with the username of the Help Desk person and replace the value of **hdpwd1** with the unencrypted password for the Help Desk person.

Now that the password file is created, add all the other Help Desk users with the command:

```
# /usr/HTTPServer/bin/htpasswd \ /usr/HTTPServer/conf/httpd.passwd
hduser hdpwd
```

Again, replace the values of **hduser** and **hdpwd** appropriately.

Tip: if your Help Desk users already have an account on any of the AIX servers, you can copy the encrypted password from */etc/security/passwd* and put it in the http password file. If this is the case, you would not need to use the **htpasswd** command at all. Simply create a text file with the following syntax: username: password.

Here is an example of my *httpd.passwd* file:

```
hduser01:Wgqw78HDqV7BU
hduser02:qB5c.yzH/Jowo
hduser03:nctUfeAYu7QAw
hduser04:FDssjhZbPzoLw
```

Now, add the Help Desk users in the HTTPD group file.

Here is an example of my *httpd.group* file:

```
helpdesk:hduser01 hduser02 hduser03 hduser04
```

In my case, I named the group **helpdesk**, but yours can be named differently. Just remember to match the same group name as in the step above.

You must now restart Apache or IHS after these changes:

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

Installation of TCL, TK, and Expect

Expect is an excellent programming language that allows you to automate normally interactive programs or applications such as the passwd program.

Expect requires either TCL/TK or Python for proper installation

using the RPM downloaded from IBM's site. In my case, I have chosen TCL/TK.

Here are the commands to install the appropriate software:

```
# rpm -i tcl-8.3.3-8.aix4.3.ppc.rpm
# rpm -i tk-8.3.3-8.aix4.3.ppc.rpm
# rpm -i expect-5.34-8.aix4.3.ppc.rpm
```

No configuration is required after installing this software.

Installation of openSSH

OpenSSH is the open-source version of secure shell. You can download it from Bull Freeware for any version of AIX at http://www.bullfreeware.com.

To install the package from Bull Freeware, which comes in exe format, simply follow the instructions found on Bull's site at http://www.bullfreeware.com/install_down.html.

Once the package is installed, you will have to generate the server's keys using the commands:

```
# ssh-keygen -t rsa1 -f /usr/local/ssh/etc/ssh_host_key -N  ""
# ssh-keygen -t rsa -f /usr/local/ssh/etc/ssh_host_rsa_key -N  ""
# ssh-keygen -t dsa -f /usr/local/ssh/etc/ssh_host_dsa_key -N  ""
```

You may have to change the path where your server's keys will be stored (ie */usr/local/ssh/etc*).

Installation of SUDO

SUDO is an application that allows the system administrator to give certain privileges to normal users in a secure fashion (without giving them privileged user passwords). All actions are logged so there is a trace of exactly what users do with SUDO.

To install SUDO, issue the command:

```
# rpm -i sudo-1.6.7p5-2.aix5.1.ppc.rpm
```

Now, you must configure SUDO so that our Web application user will be able to run privileged commands on this and every other server.

First create a new AIX group that will contain only the passwordless SSH user, **hduser**. I have chosen to name the group **hdgroup** with a gid of **522**:

```
# mkgroup -'A' id='522' hdgroup
```

Next, create the user **hduser** with a uid of **673** and a shell of **csh**:

```
# mkuser id='673' pgrp='hdgroup' groups='hdgroup' shell='/usr/bin/csh'
home='/home/hduser' \
gecos='Helpdesk Application User' hduser
```

Note 1: ensure that you use csh as this user's shell: ksh causes strange characters to appear in your Web browser.

Note 2: do not give this user a password in AIX. Authentication will be done using SSH.

Finally, add the following lines to the */etc/sudoers* file with the help of the command **visudo**:

```
Cmnd_Alias      SU_HELPDESK = /usr/bin/su - gads -c *
nobody          ALL = NOPASSWD: SU_HELPDESK
```

Set-up of passwordless SSH login for user hduser

In order to allow the user **hduser** to log in to all servers without a password, you must generate the private/public key pair.

First, as **root** change user to **hduser** and in the hduser's home directory type:

```
# su - hduser
# ssh-keygen -t dsa -f .ssh/id_dsa
```

A password will be requested; do not enter a password, just press *Enter*.

Now, go the *.ssh* directory, and you will find two new files – *id_dsa* and *id_dsa.pub*.

The second one is the public key. Rename it so:

```
# cd .ssh
# mv id_dsa.pub authorized_keys2
# chmod 600 authorized_keys2
```

Now, try to log in to the same server using ssh to verify that things worked.

Note 1: the first time you log in to a server using ssh, even though it is passwordless, it will ask you if you want to permanently add that host to the list of known hosts. You must answer **yes**.

The next time you log in to the remote server, no password will be requested nor question asked.

Note 2: this system will work as long as none of the machines changes its IP address. If this server changes its IP address, you will have to regenerate the public/private keys. If any of the other servers change their IP address, you will have to manually delete them from the *known_hosts* file in the *.ssh* directory and answer **yes** to permanently add that host to the list of known hosts again.

Installation of application files

Here are the list of application files you will need to copy into your DocumentRoot. In my case it is in the directory */usr/HTTPServer/htdocs/en_US*.

Use these commands to set the proper ownership and permissions:

```
# chown nobody app_top.php autopasswd.php blacklist index.html
pwdtools.css waiting.html serverlist
# chmod 600 app_top.php autopasswd.php blacklist index.html pwdtools.css
waiting.html serverlist
```

HTML page index.html

This page is the main entry page that users will come in through once they have authenticated using IHS or Apache. This page calls two other Web pages in frames.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Helpdesk Password Application</title>
<link href="style.css" rel="stylesheet" type="text/css">
```

```
</head>
<frameset border=0 frameSpacing=0 rows=135,* frameborder="no">
<frame border=0 frameborder="no" name=top marginWidth=5 marginHeight=5
src="app_top.php" noshade>

<frame border=0 frameborder="no" name=view marginWidth=10 marginHeight=5
src="waiting.html" noshade>
</frameset>
</html>
```

PHP page app_top.php

This PHP page is where the Help Desk user selects the server and enters the username and action. Basic verification is done once the user hits the submit button to ensure that all the required fields have data in them.

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<title>Helpdesk Password Application</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script language="javascript">
function checkrequired(which) {
  if ( which.host.selectedIndex==0 ) {
      alert("You must select a server before hitting the submit
button.");
        return false;
  }

  if ( which.user.value=='' ) {
      alert("You must enter a username before hitting the submit
button.");
      return false;
  }

  if ((which.action[0].checked==false &&
which.action[1].checked==false)) {
      alert("You must select an action before hitting the submit
button.");
        return false;
  }
  return true;
}
</script>
</head>

<body>
<table border=0 cellpadding=0 cellspacing=0 width=600>
```

```
<tr><td align=left valign=top colspan=2>
<h2>Helpdesk Password Application</h2><br></td></tr>

<form action="autopasswd.php" method="post" onsubmit="return
checkrequired(this)" target="view">

<br>
<tr><td align=left width=400>

<?php

// get the list of servers
$hostlist = file ("./serverlist");

// now sort the array of hostnames alphabetically
asort ($hostlist);
reset ($hostlist);

print "  <b>Server:</b> <select size=1 name=host>\n";
print "<option value=noneselected selected> Select a server </
option>\n";
while (list ($key, $val) = each ($hostlist)) {
      print "<option value=$val> $val </option>\n";
}
print "</select>\n";
?>

       
<b>Username:</b> <input type=text name=user size=8 maxlength=8><br><br>
  <input type=submit value="Submit"> <input type=reset
onclick="parent.view.location='waiting.html'">

</td><td align=left width=200 valign=top>
<input type=radio name=action value=r> <b>reset password</b><br>
<input type=radio name=action value=fc> <b>reset failed count</b>
</form>

</td></tr>
</table>

</body>
</html>
```

HTML page waiting.html

This HTML page is basically a blank page. Its purpose is simply to complete the bottom frame when an action is not taking place (password reset or failed count reset).

```
<html>
<head>
<title>Helpdesk Password Application</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>

 

</body>
</html>
```

Cascading Style Sheet page style.css

This page changes the look and feel of the application. Feel free to modify the colours and fonts to your preference.

```
a {text-decoration:none;}
a:link    {color:blue;  background:#ffffee;}
a:visited {color:blue;  background:#ffffee;}
a:hover   {color:black; background:#ffcc33;}
a:active  {color:black; background:#ffcc33;}
a.nobg       {text-decoration:none;}
a.nobg:link    {color:blue;  background:#ffffee;}
a.nobg:visited {color:blue;  background:#ffffee;}
a.nobg:hover   {color:black; background:#ffffee;}
a.nobg:active  {color:black; background:#ffffee;}
body{
font-size: 8pt; color: #000000;
font-family: Arial,Verdana,Helvetica,sans-serif;
font-style: normal;
background: #ffffee;
scrollbar-3dlight-color: #000000;
scrollbar-arrow-color: #000000;
scrollbar-base-color: #ffffcc;
scrollbar-darkshadow-color: #000000;
scrollbar-face-color: #ffffcc;
scrollbar-highlight-color: #ffffcc;
scrollbar-track-color: #ffffee;
scrollbar-shadow-color: #000000;
}
table{
font-size: 8pt; color: #000000;
font-family: Arial,Verdana,Helvetica,sans-serif;
font-style: normal;
}
form{
font-size: 8pt; color: #000000;
font-family: Arial,Verdana,Helvetica,sans-serif;
```

```
font-style: normal;
}
select{
font-size: 8pt; color: #000000;
font-family: Arial,Verdana,Helvetica,sans-serif;
font-style: normal;
}
textarea{
font-size: 8pt; color: #000000;
font-family: Arial,Verdana,Helvetica,sans-serif;
font-style: normal;
}
input{
font-size: 8pt; color: #000000;
font-family: Arial,Verdana,Helvetica,sans-serif;
font-style: normal;
}
option{
font-size: 8pt; color: #000000;
font-family: Arial,Verdana,Helvetica,sans-serif;
font-style: normal;
}
th {background: #990000; color: #ffffff;}
h1 {color:#990000}
h2 {color:#990000}
h3 {color:#990000}
h4 {color:#990000}
h5 {color:#990000}
h6 {color:#990000}
font.colortext{
font-size: 8pt; color: #990000;
font-family: Arial,Verdana,Helvetica,sans-serif;
font-style: normal;
}
```

PHP file autopasswd.php

This PHP page is called once the Help Desk user hits the submit button. Depending on the action requested, a different command is executed.

If the Help Desk user requests a password reset, the username entered is validated against a ‘blacklist’. The blacklist is simply a text file with the names of the users that the Help Desk is not permitted to reset passwords for. For example, root account and other generic or application accounts would not be allowed. An example blacklist file is shown below.

The password is reset to **changeme**. Once the user logs in, AIX will force the user to change their password.

```
<?php

ob_implicit_flush();

print "<html><head>\n";
print "<title>Helpdesk Password Application</title>\n";
print "<link href=\"style.css\" rel=\"stylesheet\" type=\"text/
css\">\n";
print "</head><body>\n";


// Exit if we don't get all variables we expect
if ( $user == '' || $action == ''  || $host == '' ) {
      print "Error in script usage - exit!<br>\n";
      print "</body></html>\n";
      exit;
}

// Lower case the username (windows people may be using this app :Ø )
$user = strtolower($user);

// Check whether $user is in the blacklist and
//  we want to do a password reset.
// If so, give warning and exit
$blacklist = file ("./blacklist");

// note: blacklist array has a new line character
//       in every value of array so we need to do
//       an in_array search using a new line "\n" at the end of $user
if (in_array("$user\n", $blacklist) && $action == "r") {
      print "<font color=red><b>Error: impossible to reinitialize the
account of $user.</b></font><br><br>\n";
      print "Generic user accounts are not permitted.<br>\n";
      print "Contact a system administrator for help.<br>\n";
      print "</body></html>\n";
      exit;
}

// we need at least 4k of data before it will output something
print "<font class=colortext>\n";
$char="#";
for($i = Ø; $i < 8Ø; $i++){
      print "$char";
} // for

print "<br>Please be patient while we execute the appropriate
command...<br>\n";
```

```
for($i = 0; $i < 80; $i++){
      print "$char";
} // for

print "</font><br>\n";
print "<pre>\n";

if ( $action == "r" ) {
      print "Resetting the password for $user to 'changeme'\n";
      print "on the server $host...\n\n";
      system("/usr/local/bin/sudo /usr/bin/su - hduser -c /home/hduser/
autopwd $host $user changeme", $rc);
} else {
      print "Resetting failed count for $user on the server $host...\n";
      system("/usr/local/bin/sudo /usr/bin/su - hduser -c /usr/local/
bin/ssh $host /usr/local/bin/sudo chsec -f /etc/security/lastlog -a
\"unsuccessful_login_count=0\" -s  '$user' 2>&1", $rc);
}

print "</pre>\n";

if ( $rc == "0" ) {
      print "<font color=green><b>Command completed successfully.</b></
font>";
} else {
      print "<font color=red><b>Error: command failed.</b></font>";
}

print "</body></html>\n";
?>
```

Text file blacklist

This is a sample blacklist file. It contains the names of user accounts that the Help Desk is not permitted to reset passwords for. Its contents will depend on you and your company's policies. You must enter a single username per line; no blank lines are allowed.

```
root
nobody
oracle
lpd
daemon
uucp
app
hduser
```

Text file serverlist

This is a sample serverlist file. It contains the names of all the servers that the Help Desk is permitted to reset passwords on. You must enter a single hostname per line; no blank lines are allowed.

```
server1
server2
server3
server4
```

Installation of hduser files

The Expect script used to reset the user's password is installed in the *hduser* home directory.

Set the ownership and permissions so:

```
# chown hduser:hdgroup autopwd
# chmod 770 autopwd
```

Expect script autopwd

This Expect script is used to reset the password of an end user. It spawns a Korn shell, issues a **ssh** to the server requested, and subsequently issues the **passwd** command. The username typed is entered, and the new, temporary password, **changeme**, is entered twice.

If at any point during the **passwd** operation an error occurs, an error message will appear in the browser.

```
#!/usr/bin/expect
#

set force_conservative 0  ;# set to 1 to force conservative mode even if
                     ;# script wasn't run conservatively originally
if {$force_conservative} {
       set send_slow {1 .1}
       proc send {ignore arg} {
             sleep .1
             exp_send -s -- $arg
       }
}

set timeout -1
```

```
spawn /usr/bin/ksh
set host [lindex $argv 0]
set username [lindex $argv 1]
set password [lindex $argv 2]
match_max 100000
set prompt "(%|#|>|\\$) $" ;# default prompt
catch {set prompt $env(EXPECT_PROMPT)}
expect -re $prompt
send "/usr/local/bin/ssh $host\r"
expect -re $prompt
send "/usr/local/bin/sudo /usr/bin/passwd $username\r"
expect {
      -re $prompt {
            send_user "\n\n!!! ERROR DURING PASSWORD RESET FOR $username
!!!\n"
            exit 1
      }
      "password:"
}
send "$password\r"
expect "new password"
send "$password\r"
expect -re $prompt
send -- "exit\r"
expect "closed."
send -- "exit\r"
expect eof
```

hduser .cshrc file

This is the *.cshrc* file for the user **hduser**. If you change the prompt variable, you risk breaking the Expect script. I set a large history file so that we can go back and see everything this user has executed over a long period of time.

```
set prompt="`/usr/bin/whoami`@`/usr/bin/hostname`$ "
set path = ($path /usr/local/bin /usr/bin /etc /usr/sbin /usr/ucb $HOME/
bin /sbin .)
set history=5000
set savehist=5000
```

CONFIGURATION ON ALL THE OTHER AIX SERVERS

On all the servers that you wish to allow the resetting of passwords and a failed login count, you must install the following:

1 OpenSSH – see above.

2 SUDO – see above.

3 Create the AIX group **hdgroup**, and user **hduser** as outlined above.

4 Once the **hduser** account is created, copy the following files from the application Web server to the new server:

   – */home/hduser/.cshrc*

   – */home/hduser/.ssh/authorized_keys*.

5 Change the permissions for the local **hduser** account so:

```
# chown -R hduser:hdgroup /home/hduser
# chmod -R 700 /home/hduser
# chmod 600 /home/hduser/.ssh/authorized_keys
```

6 Add the following lines in SUDO using the command **visudo**:

```
Cmnd_Alias      CHSEC = /usr/bin/chsec
Cmnd_Alias      PWD = /usr/bin/passwd
hduser          ALL = NOPASSWD: CHSEC, PWD
```

7 From the application Web server, run a **ssh** to this new server with the **hduser** account to ensure a passwordless entry occurs. Remember, you will be asked whether you want to permanently add the new host to the *known_hosts*. You must enter **yes**.

8 Add the name of this server in the *serverlist* file. In my case, it is in the directory */usr/HTTPServer/htdocs/en_US/* – see above.

TESTING

You are now ready to test the Web application. Follow these steps:

1 Open a Web browser and type in the URL you entered as your **ServerName** in the *httpd.conf* file. In my case it was helpdesk.domain.com.

2 You will be prompted for a **username** and **password**. Enter the one that you added to the *httpd.passwd* file earlier. In my case it was username **hduser01** and password **hdpwd1**.

3 You will now be presented with a selection screen. Test all possibilities including users that should be blacklisted.

4 Once the action completes, you should have a message saying it was successful or it failed.

5 When resetting the **failedcount** for a user, the blacklist is not verified because it could not cause any harm.

GOTCHAS

Here are some things to watch out for and verify if things don't work as expected:

1 The **hduser** should not have a password, so if someone tries to log in as this user, the **failedcount** could go beyond three very quickly. If this occurs, SSH will ask for a password even if it finds a key. As a workaround, reset the **failedcount** for the **hduser** or set it to unlimited for this user.

2 If passwordless ssh is causing you problems, run the client in debug mode so:

```
ssh -v -v -v hostname
```

The more **-v**, the more verbose. If you still can't find the problem, run the server **ssh** process in debug mode as well. Use:

```
sshd -d -d -d
```

The more **-d**, the more verbose.

3 If you want to get rid of the MOTD banner when using SSH, go into the *sshd_config* file and set the line **PrintMotd** to **no**. Restart **sshd**.

Figure 1: Screenshot of application in use

4 If any of your servers change IP address, remember to remove the old line in the *known_hosts* file and answer **yes** when asked to add it back.

SCREENSHOT

Figure 2 shows an example of the screen that users of the application would see.

Elvio Prattico
Consultant
PRATTICO Consulting Inc (USA)

AIX sets new record

Last November IBM set a new record with a TPC-C benchmark result of 3,210,540 tpmC at a price/performance of $5.19/tpmC. And if that doesn't make any sense to you, IBM is claiming that its performance is 2.7 times better than Oracle RAC, its closest competitor. In fact, the new result beats the previous record by a margin wider than the top results of Oracle and Microsoft, the next closest competitors, combined. So how did IBM get this kind of scalability? By using DB2 Universal Database Version 8.2, a 64-processor eServer p595 server running AIX (of course), and TotalStorage FASt900 storage. And if you're wondering what TPC-C is, well, it's an industry standard benchmark test for measuring performance and price/performance of systems in an on-line transaction-processing environment.

For those of you wanting more details about the results, the benchmark simulated 2.56 million users of DB2 UD, the database contained more than 100 billion rows, and the system managed 240TB of storage and 2TB of random access memory.

IBM was particularly pleased because earlier in the year it had published TPC-C results on similar hardware in smaller configurations (an eight-processor server and a 16-processor server). Both had scored well. The 64-processor results were very good in terms of scalability. The 64-processor DB2 UDB result delivered 99% of linear scalability when compared with the 16-processor result. Also highlighted by IBM is the cost or price/performance scalability demonstrated by its results. Even though the 64-processor configuration yielded almost four times the performance of the 16-way result, the price/performance only increased by 5%, giving DB2 the two best price/performance results among the top 10 overall performance results.

You can get more information on the top 10 TPC-C results by

performance (tpmC) from www.tpc.org/tpcc/results/tpcc_perf_results.asp?resulttype=all.

Exploring AIX system identification facilities on AIX 4 and AIX 5L

Unique system identifiers are used by IBM customers, OEMs, and ISVs in order to implement inventory tracking, maintenance, and licensing. It was a relatively simple item of information when the first uniprocessor RS/6000 systems appeared. However, with the recent introduction of large massively-partitioned machines, a need to extend this facility has occurred. This article will present the facilities available to retrieve this information as well as methods to understand its various components.

AIX SYSTEM IDENTIFICATION RETRIEVAL COMMANDS AND APIS

AIX provides several methods to collect the system identification information, which can include part or all of the following elements:

- The machine sequence number
- The manufacturing plant code
- The machine type
- The machine model
- The partition number if executed in LPAR/DLPAR/micro-partitioning environment.

System APIs **uname()** and **unamex()** allow the retrieval of system identification as part of the structures defined by types utsname and xutsname. The field names are machine and nid

respectively. AIX 5.3 adds a new file to the xutsname called longnid.

We will use the following minimalist program to demonstrate the use of APIs.

For AIX 4.3, 5.1, and 5.2:

```
#include <sys/utsname.h>

main(){
struct utsname un;
struct xutsname xun;

uname(&un);
        printf("sysname: %s nodename: %s release: %s version: %s
machine: %s\n"
un.sysname, un.nodename, un.release, un.version, un.machine);
unamex(&xun);
printf("nid: %X \n",xun.nid);
}
For AIX 5.3:
#include <sys/utsname.h>

main(){
struct utsname un;
struct xutsname xun;

uname(&un);
printf("sysname: %s nodename: %s release: %s version: %s machine: %s\n"
un.sysname, un.nodename, un.release, un.version, un.machine);
unamex(&xun);
printf("nid: %d longnid: %llX\n",xun.nid,xun.longnid);
}
```

We will call this program **ut**.

It is also possible to retrieve this information by using the command **lsattr**:

```
# lsattr -El sysØ -a systemid
```

The command **uname** produces a variety of information about the computer on which it is executed. Flags relevant for the retrieval of the system identifications are:

- uname -m – produces the machine sequence number, considered obsolete and not recommended.

- uname -u – produces plant code and machine ID.
- uname -M – produces type and model of machine.
- uname -L – shows the partition number and ID.

SYSTEM IDENTIFICATION RETRIEVAL UNDER AIX 4.3, 5.1, AND 5.2

We will demonstrate and decipher the information that is retrieved by the methods described above for pre-AIX 5.3 systems. We will use the following machine for our demonstration – a Model 7025-F50 running AIX 4.3 (no LPARs of course).

We will give the command followed by the output on 7025-F50:

- **./ut**

  sysname: AIX nodename: ibmf50 release: 3 version: 4 machine: 0043F97A4C00

  nid: 43F97A4C

- **lsattr -El sys0 -a systemid**

  systemid IBM,01443F97A Hardware system identifier False

- **uname -u**

  IBM,01443F97A

- **uname -M**

  IBM,7025-F50

- **uname -L**

  uname: Not a recognized flag: L

  Usage: uname [-snlrvmaxuMS:T:]

- **uname -Mu**

  IBM,7025-F50 IBM,01443F97A

- **uname -MuL**

  uname: Not a recognized flag: L

  Usage: uname [-snlrvmaxuMS:T:]
- **uname -m**

  0043F97A4C00
- **uname -F**

  uname: Not a recognized flag: F

  Usage: uname [-snlrvmaxuMS:T:]
- **uname -f**

  uname: Not a recognized flag: f

  Usage: uname [-snlrvmaxuMS:T:]

As can be seen, the five-digit sequence produced by **uname -m** as well as by our **ut** program is not sufficient to identify the computer in a unique way.

The best option for the unique identification of computers running under AIX 4.3, 5.1, and 5.2 without logical partitions is to use the output of the **uname -Mu** command. This information can be broken into several data items:

- Machine Type – 7025
- Machine Model – F50
- Common prefix – IBM,01
- Plant Code – 44 (Santa Palomba)
- Sequence Number – 3F97A.

AIX SYSTEM IDENTIFICATION RETRIEVAL UNDER AIX 5.3 AND PATCHED 5.1/5.2

Following the introduction of partitioning, providing the ability to split computers into up to 32 partitions on POWER4 servers

and up to 254 partitions on POWER5 servers, the requirement for the generation of unique system identifiers has greatly increased.

It has been decided to base the generation of unique system identifiers on the following data:

- Four-digit numeric machine type
- Three-digit alphanumeric model name
- Two-digit alphanumeric manufacturing plant designation
- Five-digit alphanumeric sequence number.

A method has been developed to construct unique system IDs by using one or more elements from the above list.

Two new flags have been added to the **uname** command:

- **-F** – displays a system identification string comprising hexadecimal characters. This identification string is the same for all partitions on a particular system.
- **-f** – similar to the **F** flag, except that the partition number is used in the calculation of this string. The resulting identification string is unique for each partition on a particular system.

The definition of the **-m** flag of the **uname** command has been changed: **-m** now displays the machine ID number of the hardware running the system.

Note: the **-m** flag cannot be used to generate a unique machine identifier for partitions in an LPAR environment.

Compatibility with earlier environments is preserved in the output of the following flags of **uname** command: **-M**, **-u**, and **-L**.

The API defined in the header file */usr/include/sys/utsname.h* has been changed as well. The xutsname structure, which was defined before as:

```
struct xutsname {
```

Command	*Output on 7028-6C4*	*Output on 7040-671*
./ut	sysname: AIX nodename: ibmp630 release: 3 version: 5 machine: 005730DA4C00 nid: 1462819404 longnid: 36E8374B58B9E01	sysname: AIX nodename: ibmp6701 release: 2 version: 5 machine: 0029B01C4C00 nid: 29B01C4C
lsattr –El sys0 –a systemid	systemid IBM,0165730DA Hardware system identifier False	systemid IBM,01029B01C Hardware system identifier False
uname –u	IBM,0165730DA	IBM,01029B01C
uname –M	IBM,7028-6C4	IBM,7040-671
uname –L	1 NULL	1 ibmp6701
uname -Mu	IBM,7028-6C4 IBM,0165730DA	IBM,7040-671 IBM,01029B01C
uname –MuL	IBM,7028-6C4 IBM,0165730DA 1 NULL	IBM,7040-671 IBM,01029B01C 1 ibmp6701
uname -m	005730DA4C00	0029B01C4C00
uname -F	036E8374B58B9E01	3700008EE7DF000
uname -f	036E8374B58B9E01	3700008EE7DF001

Figure 1: Output comparison

```
        unsigned int nid;
        int reserved[3];
};
```

has been changed to:

```
struct xutsname {
        unsigned int nid;
        int reserved;
          /* Added in order to preserve structure size and alignment */
        unsigned long long longid;
};
```

APARs IY52116 and IY52125 have been introduced for AIX 5.1 and 5.2 respectively, in order to upgrade computers running these versions of the AIX system to use the new system identification generation method.

We will demonstrate the information that is retrieved by the methods described above for systems running AIX 5.3 and patched AIX 5.2. We will use the following machines for our demonstration:

- Model 7028-6C4 running AIX 5.3 (not partitioned)
- Model 7040-671, running AIX 5.2 (executed in partition 1).

The command and the output on 7028-6C4 and the output on 7040-671 are shown in Figure 1.

The decoding of the information for these machines is:

- Machine Type – 7026/7040
- Machine Model – 6C4/671
- Common prefix – IBM,01/IBM,01
- Plant Code – 65 (Dublin/93B)/02 (Poughkeepsie/992)
- Sequence Number – 3F97A / 9B01C
- Partition name and number – 1 NULL (no partitions) / 1 ibm6701 (first partition).

SUMMARY

In this article I have described the methods that can be used in legacy and new AIX operating system environments in order to retrieve the unique system identification data.

REFERENCES

1 *AIX System Identification*, Wane Huang and Bradford Cobb, IBM Solutions Enablement, 27 October 2004.

Alex Polak
System Engineer
APS (Israel)

The Andrew Filesystem (AFS)

AFS (the Andrew Filesystem) is a distributed filesystem that enables users to share and access all the files stored in a network of computers as easily as they access the files stored on their local machines. AFS takes advantage of the interconnected nature of the network by storing files on more than one computer in the network and making them accessible to all. The responsibility for file storage and delivery is distributed among multiple machines instead of relying on only one.

AFS uses a server/client model. File server machines store the files in the distributed filesystem, and a server process running on the file server machine delivers and receives the files. Clients provide users with access to the files stored on the file server machines.

One of the features that makes AFS easy to use is that it provides transparent access to the files in a cell's filespace. Users do not have to know which file server machine stores a particular file in order to access it; they simply provide the file's pathname, which AFS automatically translates into a machine location.

A cell is an administratively-independent site running AFS. In a cell you can make many decisions about configuring and maintaining your cell in the best way that best serves its users.

AFS groups files into volumes, making it possible to distribute files across many machines. A volume is a unit of disk space that functions like a container for a set of related files – keeping them all together on one partition. Volumes increase file availability through replication and back-up. A mount point is similar to a symbolic link in the file tree that specifies which volume contains the files kept in a directory.

The first convention is that the top level in the file tree be called the */afs* directory. The second convention is that just below the */afs* directory you place directories corresponding to each cell.

The partitions that house AFS volumes on a file server machine must be mounted on directories named */vicepindex,* where *index* is one or two lower-case letters. By convention, the first AFS partition created is mounted at the */vicepa* directory, the second at the */vicepb* directory, and so on through to the */vicepz* directory. The names then continue with */vicepaa* to */vicepaz*, */vicepba* to */vicepbz*, and so on, up to the maximum supported number of server partitions.

Before writing something in the AFS filesystem, you have to get a token. Once logged in, a user can obtain a token at any time with the **klog** command. To discard either all tokens or the token for a particular cell, issue the **unlog** command. The command affects only the tokens associated with the current command shell.

To display the tokens associated with the current command shell, issue the **tokens** command. The following examples illustrate its output in various situations.

```
# tokens

Tokens held by the Cache Manager (UID Based Tokens):

   —End of list—
```

If there is a token:

```
# tokens

Tokens held by the Cache Manager (UID Based Tokens):

User's (AFS ID 51360) tokens for afs@prodcell [Expires Dec  4 13:58]
   —End of list—
```

Below are my scripts to get/kill tokens automatically.

GET_TOKEN.SH

```
#!/bin/ksh
# Adnan Akbas, Turkcell, 03.04.2004
# This script is to get an AFS token for AFS administrator (25 hours)
# variables ###################################################################
version=1.0
passwd=turkcell2004
```

```
adm=afsadm
afs_path=/usr/afs/bin
afscell=$(cat /usr/afs/etc/ThisCell)
# main ################################################################
echo "\n\nINFO: $(date +"%H:%M %d.%m.%Y") starting script $Ø Version
$version ..."

#get token
echo "INFO: $afs_path/klog -principal $adm -password "*******" -cell
$afscell"
$afs_path/klog -principal $adm -password $passwd -cell $afscell

if [ $? != Ø ]
then
   echo "ERROR: klog not correct !!!, exit 1"
   exit 1
fi
## check whether successful
if [ $($afs_path/tokens | grep "tokens for afs@${afscell}" | wc -l) -ne
1 ]
then
   echo "ERROR: not successful to get afs token , exit 2"
   exit 2
fi
echo "INFO: $(date +"%H:%M %d.%m.%Y")  ending script $Ø "
```

KILL_TOKEN.SH

```
#!/bin/ksh
# Adnan Akbas, Turkcell, Ø3.Ø4.2ØØ4
# This script kills any AFS token
# variables ###########################################################
version=1.Ø
afs_path=/usr/afs/bin
scr_path=/usr/afs/scripts

# main ################################################################
echo "\n\nINFO: $(date +"%H:%M %d.%m.%Y") starting script $Ø Version
$version ..."

# kill tokens
echo "INFO: $afs_path/unlog"
$afs_path/unlog

if [ $? != Ø ]
then
   echo "ERROR: unlog not OK !!!, exit 1"
   exit 1
fi
```

```
echo "INFO: $(date +"%H:%M %d.%m.%Y")  ending script $Ø "
```

Here are some useful commands that I used in my scripts to manage volumes:

- Create read/write volume:

  `vos create`

- Create read-only volume:

  `vos addsite and vos release`

- Examine VLDB entry:

  `vos listvldb`

- Examine volume header:

  `vos listvol`

- Create mount point:

  `fs mkmount`

- Remove mount point:

  `fs rmmount`

- Display mount point:

  `fs lsmount`

- Move read/write volume:

  `vos move`

- Set volume quota:

  `fs setvol or fs setquota`

- Remove read/write volume:

  `vos remove and fs rmmount`

- Remove read-only volume:

  `vos remove`

- Remove VLDB entry; no volume change:

  `vos delentry`

- Unlock volume:

```
vos unlock
```

- Unlock multiple volumes:

```
vos unlockvldb
```

- Lock volume:

```
vos lock
```

The Volume Location Server maintains a complete list of volume locations in the Volume Location Database (VLDB). The VLDB includes entries for every volume in a cell. When the Cache Manager begins to fill a file request from an application program, it first contacts the Volume Location Server in order to learn which file server machine currently houses the volume containing the file.

To display the VLDB entry for one or more volumes, use the **vos listvldb** command (here the volume name is testvol with three versions, and servers server1, server2, and server3):

```
# vos listvldb

VLDB entries for all servers

testvol
    RWrite: 536870921     ROnly: 536870922
    number of sites -> 4
       server server2 partition /vicepa RW Site
       server server2 partition /vicepa RO Site
       server server1 partition /vicepa RO Site
       server server3 partition /vicepa RO Site
    Volume is currently LOCKED

testvol_040421_1708
    RWrite: 536871011     ROnly: 536871012
    number of sites -> 4
       server server2 partition /vicepa RW Site
       server server2 partition /vicepa RO Site
       server server1 partition /vicepa RO Site
       server server3 partition /vicepa RO Site

testvol_040421_1710
    RWrite: 536871014     ROnly: 536871015
    number of sites -> 4
```

```
        server server2 partition /vicepa RW Site
        server server2 partition /vicepa RO Site
        server server1 partition /vicepa RO Site
        server server3 partition /vicepa RO Site

testvol_040428_0900
    RWrite: 536871020     ROnly: 536871021
    number of sites -> 4
        server server2 partition /vicepa RW Site
        server server2 partition /vicepa RO Site
        server server1 partition /vicepa RO Site
        server server3 partition /vicepa RO Site
```

Issue the **vos create** command to create the volume:

```
# vos create <machine name> <partition name> <volume name>
                    [-maxquota <initial quota (KB)>]
```

Issue the **fs mkmount** command to mount the volume in the filespace:

```
# fs mkmount <directory> <volume name>
```

Issue the **fs rmmount** command to remove the mount point:

```
# fs rmmount <directory>
```

Issue the **fs lsmount** command to verify that the mount point refers to the correct volume:

```
% fs lsmount <directory>
```

Issue the **vos addsite** command to define each new read-only site in the VLDB:

```
# vos addsite <machine name> <partition name> <volume name or ID>
```

Issue the vos release command to clone the read/write source volume and distribute the clone to each read-only site:

```
# vos release <volume name or ID>
```

Issue the vos examine command to display the volume's current sites:

```
# vos examine <volume name or ID>
```

Instead of dealing every time with the commands above when I need a volume, I wrote some scripts to automate and ease my work.

First of all, I prefer to create a parameter file (param.dat) to define all my variables that I use in my scripts (creating variables for testvol):

```
# create_vars.sh  testvol
```

CREATE_VARS.SH

```
#!/bin/ksh
# Adnan Akbas, Turkcell, 04.04.2004
# This script is to create the parameter file for all AFS scripts.
# variables ###########################################################
version=1.0
# Assign input value to variable
myvol=$1

# Get own cellname
if [ ! -f /usr/afs/etc/ThisCell ]
then
   echo "ERROR: /usr/afs/etc/ThisCell not found !!! "
   exit 2
fi

afs_cell=$(cat /usr/afs/etc/ThisCell)
afs_path=/usr/afs/bin
afs_log=/usr/afs/logs
scr_path=/usr/afs/scripts
afs_param=$scr_path/param.dat
time_stamp=$(cat ${scr_path}/time_stamp.dat)
afs_local=/usr/afs/local
afs1=server1
afs2=server2
afs3=server3
afsparta=/vicepa
afspartb=/vicepb
afspartc=/vicepc
afspartd=/vicepd
afsrw=/afs/.${afs_cell}_rw
afsro=/afs/${afs_cell}
afsrw1=$afsrw/$myvol
afsro1=$afsro/$myvol
afsvol1=${myvol}_$time_stamp
afsvol1mnt=$afsvol1
afsrw2=$afsrw1/$afsvol1
afsro2=$afsro1/$afsvol1
other_user=system:anyuser
other_perm=rl

# main ################################################
```

```
echo "\n\nINFO: $(date +"%H:%M %d.%m.%Y") starting script $Ø Version
$version ..."

## Check the number of arguments
if [ $# -ne 1 ]
then
   echo "ERROR: The number of arguments is wrong!"
   xit 1
fi

# clear afsserver[*] variables
i=Ø
while (( i < 4 ))
do
   unset afsserver[$i]
   (( i = i + 1 ))
done
#
# structure
# AFS_Volume_name             Mountpoint
# root.afs                    /afs
# root.cell                   /afs/.prodcell_rw
# example_yymmdd_HHMM         /afs/.prodcell_rw/example/
example_yymmdd_HHMM
# abcdefgh_i_yymmdd_HHMM      1Ø characters are available
# 123456789Ø123456789Ø12      not more than 22 characters
#                                                 to name a volume
#
## only   >>=abcdefghij<<  characters are possible! (level 1)
##           1234567890

echo "INFO: set variables for AFS-Content ($myvol) ..."
case $myvol in
   testvol )      afsquota=1ØØØØØ       # in kB = 1ØØ MB
                  afsserver[Ø]=$afs2
                  afsserver[1]=$afs1
                  afsserver[2]=$afs3
                  afspart=$afsparta
                  afsbackupmin=3
                  adm_user=cmsadmin
                  adm_perm=rlidwk
                  ;;
   prodvol )      afsquota=1ØØØØØ       # in kB = 1ØØ MB
                  afsserver[Ø]=$afs3
                  afsserver[1]=$afs1
                  afsserver[2]=$afs2
                  afspart=$afspartb
                  afsbackupmin=3
                  adm_user=cmsadmin
                  adm_perm=rlidwk
```

```
                ;;
   * )          echo "ERROR: Parameter >> $1 << not defined !!"
                echo "ERROR: exit 5\n"
                exit 5
                ;;
esac

if [ -f $afs_param ]
then
   echo "INFO: delete old AFS Parameterfile..."
   rm $afs_param
fi
echo "INFO: generate new AFS Parameter file (param.dat) ..."
echo "#!/bin/ksh
#
# AFS  Parameter Script
# Name: param.dat
#
# Automatically-generated script !!
# Do not change manually !
#
# Adnan Akbas, Turkcell
#
myvol=${myvol}
afs_cell=${AFSCELL}
afs1=${afs1}
afs2=${afs2}
afs3=${afs3}
AFS4=${AFS4}
afsserver[Ø]=${afsserver[Ø]}
afsserver[1]=${afsserver[1]}
afsserver[2]=${afsserver[2]}
afsserver[3]=${afsserver[3]}
afspart=${afspart}
afsbackupmin=${afsbackupmin}
time_stamp=${time_stamp}
afs_path=${afs_path}
afs_log=${afs_log}
scr_path=${scr_path}
afs_param=${afs_param}
afs_local=${afs_local}
afs_cell=${afs_cell}
afsrw=${afsrw}
afsro=${afsro}
afsrw1=${afsrw1}
afsro1=${afsro1}
afsrw2=${afsrw2}
afsro2=${afsro2}
afsvol1=${afsvol1}
afsvol1mnt=${afsvol1mnt}
```

```
other_user=${other_user}
other_perm=${other_perm}
adm_user=${adm_user}
adm_perm=${adm_perm}

" > $afs_param
echo "INFO: $(date +"%H:%M %d.%m.%Y")  ending script $Ø "
```

Editor's note: this article will be concluded next month.

Adnan Akbas
Senior System Administrator
TURKCELL (Germany)

AIX sites that also have Sun will be interested to know that Sun Microsystems has announced plans to acquire SevenSpace, which provides applications that remotely monitor enterprise applications, databases, and network devices across a number of platforms and technologies.

With this acquisition, Sun has expanded its capabilities from simply monitoring Sun technology to monitoring and managing technologies from several vendors, including applications that run on AIX, HP-UX, Windows, and Red Hat Linux platforms.

For further information contact:
URL: www.sun.com/smi/Press/sunflash/2004-11/sunflash.20041129.1.html.

* * *

Relicore has announced Version 4.0 of Relicore Clarity, its automated IT service configuration management software. The new version includes advancements in the areas of scalability, enterprise integration, platform coverage, and product functionality, providing customers with IT service configuration management capabilities that are the foundation for IT Infrastructure Library (ITIL)-based IT Service Management (ITSM) best practices.

Highlights of Relicore Clarity V4.0 include: global mapping, enabling users to automatically discover, map, and view applications, servers, and related dependencies across their servers; enhanced integration with third-party enterprise management systems, including those from HP, IBM, CA, and BMC; dependency characterization, allowing users to drill-down into application and server dependencies and view details such as when the dependency was established, its current state, and frequency of communication; and consistency management, enabling users to ensure that servers and applications remain consistent.

Version 4.0 now supports AIX and SUSE Linux.

For further information contact:
URL: www.relicore.com/products.

* * *

Micromuse has announced Version 7 of Netcool/OMNIbus, which provides a strategic service assurance platform and the foundation for operations management.

Greater event handling power in Version 7 allows users to more-efficiently access, manipulate, visualize, and report on the raw data. The new version also adds intelligent event reduction with advanced procedural language and database triggers to allow for batch processing and more complex data manipulations, which facilitate business service management and service quality management.

The software runs on AIX 5.2, Red Hat Enterprise Linux AS, ES, and WS 2.1, and Windows 2000 Server, Advanced Server, and 2003 Server.

For further information contact:
URL: www.micromuse.com/news/press/pressview.cgi?&article=v76-Dec-2004.

* * *