MIGRATING SUN JAVA SYSTEM MESSENGER EXPRESS PERSONAL ADDRESS BOOK

USING THE PAB2ABS.PL UTILITY

Sarma Vempati, Collaboration Products

Sun BluePrints™ OnLine — December 2006

Part No 820-0618-00
Revision 1.0, 12/8/06
Edition: December 2006

Table of Contents

Migrating Sun Java System Messenger Express Personal Address Book

The Messenger Express Web-based email client (both Sun Java™ Enterprise System and iPlanet™ versions) includes a Personal Address Book (PAB) application for storing and managing user's personal information, such as email addresses and phone numbers. Sun Java System Communications Express, the unified Web client introduced in Sun Java Enterprise System 2004Q2 supersedes Messenger Express and Calendar Express. Communications Express also includes Address Book Store (ABS) that provides all of the functionality of PAB and is better integrated with mail and calendar components.

When upgrading from Messenger Express (also known as Webmail) to Communications Express, you need to migrate users' PAB entries to ABS. (This migration does not occur automatically as part of the upgrade process.) Unfortunately, the `runMigrate.sh` utility, provided for the PAB migration, runs slowly with a large PAB database. Thus, a new tool, `pab2abs.pl`, has been made available, which addresses the performance problems of `runMigrate.sh`. As described in this article, you can use the `pab2abs.pl` tool to either migrate a single or a few users, or to migrate your entire PAB database.

How This Article Is Organized

This article addresses the following topics:

Conceptual understanding and familiarity with Messaging Server and Directory Server administration is valuable in developing the right strategy for the PAB migration in large scale deployments. You need administrative privileges to try the examples provided, or to execute the migration.

Overview of the Issues

Both PAB and ABS store personal information in an LDAP directory server as attribute value pairs. Their schemas differ, however, and ABS contains a larger set of attributes. In Java Enterprise System 2005Q4 (Java ES 4), Messenger Express and Calendar Express are deprecated in favor of Communications Express as the mail, calendar and address book client. As you upgrade from Messenger Express to

Communications Express, you need to migrate the Messenger Express PAB to provide continuity with users' valuable personal information. The `runMigrate.sh` script, available within Communications Express, enables you to migrate PAB information in either batch and dynamic modes. However, experience has shown that the `runMigrate.sh` script runs slowly when migrating a large PAB database. Analysis of `runMigrate.sh` revealed that the script retrieves user PAB entries one at a time, generates ABS entries, then adds the information to the personal store (ABS directory server) by using LDAP. Transformation of millions of entries, one-by-one over, the network, is very time consuming and consequently migration of a large PAB database has been known to take several days (or even a few weeks).

Noting that Sun Java System Directory Server bulk export and import are extremely fast compared to modification or creation of entries over LDAP, Sun has created a new PAB migration tool, `pab2abs.pl`. Testing of `pab2abs.pl` in-house, as well by certain customers, shows an impressive reduction of PAB migration times, by a factor of 30 or more. Thus, Sun is recommending to customers to use the `pab2abs.pl` tool in place of the `runMigrate.sh` script.

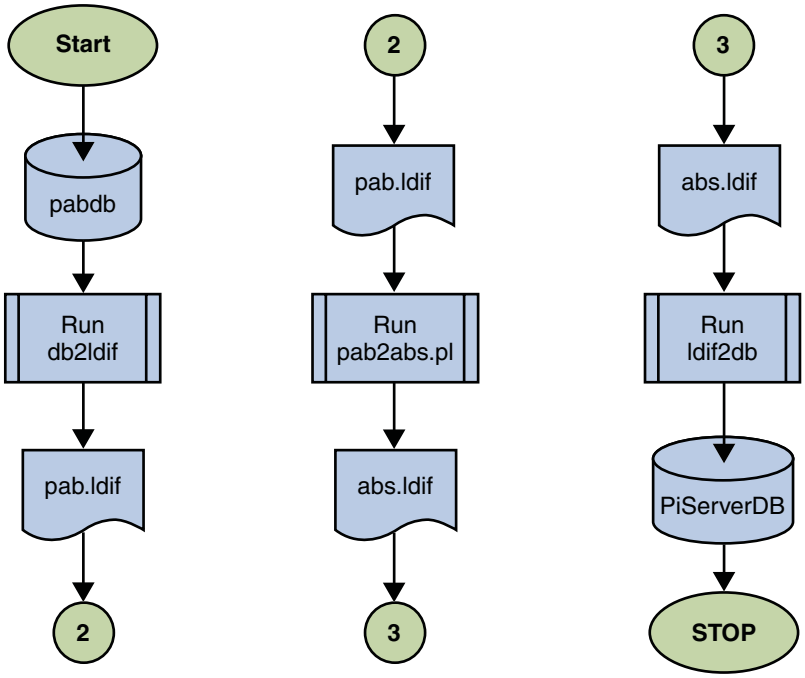
Overview of PAB Migration

The high-level overview of migrating PAB data to ABS involves three steps:

- Exporting PAB database as LDIF file
- Generating ABS database in LDIF by using `pab2abs.pl` (takes above file as input)
- Importing ABS LDIF (initialize the `o=PiServerDB`)

Figure 1 on page 3 represents the three stages of this migration process.

Figure 1. Three Stages of PAB to ABS Migration



Details About the Migration Tool

The `pab2abs.pl` tool generates a new LDIF file, translating the PAB schema from the input LDIF file to that of ABS. (See Table 1 on page 4 for more information.) The `pab2abs.pl` tool translates only the attribute names and leaves values unchanged, thus maintaining data integrity. As a prerequisite, attribute values must not be folded in the input PAB LDIF file. Furthermore, the separator between names and values must be ‘ :’ (colon). LDAP V1 syntax is not supported.

Functional Requirements

The `pab2abs.pl` tool applies the following logic:

- Generates object classes to support the ABS schema every PAB DN.
- Translates only the first occurrence of an attribute. Second and subsequent occurrences of the same attribute are ignored. Such exceptions may have resulted from disabling PAB schema checking.
- Does not distinguish from short and long attribute names, for example `cn` and `commonName`. Transforms only the first occurrence.
- Transforms all attributes containing a language subtype, that is, `cn|givenName|sn`.
- Translates multiple values of the `mail` attribute, though an inconsistency, to `work`, `home`, and so on. The `mail` attribute is a single-valued attribute per schema.
- Properly links non-contiguous contact and group entries of a user’s address book. Neither bulk `ldapsearch` nor `db2ldif` generate all entries of a user’s PAB contiguously. Rather, they appear in chronological order. An example of this serious exception is the case of the pab user dn (`ou=user1,ou=people,o=red.example.com,o=jes3`) followed by contacts rather than by the

address book DN (`dn: un=AddressBookc5b9cab, ou=user1, ou=People, o=red.example.com, o=jes3, o=pab`).

- `PiEntryID` is set to the unique identifier (`un`) value of the PAB entry. Valued for crossmatching PAB and ABS entries readily, `PiEntryID` is preferred by some customers. The `pab2abs.pl` tool has the ability to generate unique hex digits for `PiEntryID`.
- Generates user root modifications in support of ABS in a separate file (see “Options” on page 5 for more information).
- Logs translation errors.

An inconsistency noted in customer PAB databases is the absence of the object class `pab`. Upon login, a new PAB is created for such a user, and previous contacts are orphaned. The orphaned contacts are also faithfully translated.

Another inconsistency detected in customer data is the occurrence of multiple `pab` object classes, possibly due to Directory Server replication issues. You should check whether the first `pab` migrated corresponds to the active address book running `pab2abs.pl`. These cases are flagged as errors in the Directory Server `errors` log file during the third stage of the migration process.

The `pab2abs.pl` tool can accommodate variations in PAB DN and PSRoot. Options for output, logging debug information, and error logging, are also provided. See “Options” on page 5 for more information.

Attribute Translation Table

The following table shows how the `pab2abs.pl` tool migrates PAB attributes to ABS attributes. The first column shows the set of attributes available in Messaging Server PAB, which is a subset of those supported by PAB Schema. The `pab2abs.pl` tool translates PAB attributes (first column) to ABS attributes (second column). The value is carried over in most of the cases unchanged, with the exception of special characters and multi-line/multi-attribute values, which are base64 encoded. The third column provides additional notes concerning the translation.

Table 1. Attribute Transformation

PAB	ABS	Comments
`un`	`piEntryID`	Tool assigns PAB value to ABS, no new hex number generated.
`givenName`	`givenName`	
`description`	`multiLineDescription`	
`sn, surname`	`sn`	One time only.
`cn, commonName`	`cn`	One time only.
`mail`	`PiEmail1, PiEmail1Type: work`	First occurrence.
`mail`	`PiEmail2, PiEmail2Type: home`	Second occurrence.
`mail`	`PiEmail3, PiEmail3Type: other`	Third occurrence.
`rfc822mailbox`	`nil`	Ignored (same as mail).

PAB	ABS	Comments
telephoneNumber	piPhone1, piPhone1Type: work	
homePhone	piPhone2, piPhone2Type: home	
mobile	piPhone3, piPhone3Type: mobile	
pager	piPhone4, piPhoneType: pager	
facsimileTelephoneNumber	piPhone5, piPhone5Type: fax	
street	homePostalAddress	
l (location)	homeCity	
st (state)	homeState	
postalcode	homePostalCode	
co (country)	homeCountry	
labeldURI	piWebSite	
dateOfBirth	dateOfBirth	
memberOfPabGroup	memberOfPIGroup	

pab2abs.pl Command Description

This section describes the command syntax and options for the `pab2abs.pl` tool.

Syntax

```
pab2abs.pl {-b pabdn -d domain -h hosts [-p port] | -f file} -i file [-a file] [-l file] [-o file] [-v | -r]
```

Options

The options are:

Option	Description
	`pabdn` specifies the base DN of the Personal Address Book. White space surrounding components is not permitted.
`-b` *pabdn*	Valid example: `ou=people,ou=red.siroe.com,ou=jes3`
	Invalid example: `ou=people, ou=red.siroe.com, ou=jes3`
`-d` *domain*	Typically corresponds to the DNS domain name. ABS entries from a domain are created under the tree `o=`*domain*`,o=PiServerDB`.
`-h` *hosts*	Directory Server host names (FQDN).
`-p` *port*	Port on which Directory Server is running. The default is 389.

Option	Description

`-f` *file*	Specifies a file that contains the first four parameters. Use ; (semi-colon) to separate parameters and values. Format:
	
	`domain: red.example.com`
	`pabdn: ou=people,o=red.example.com,o=jes3,o=pab`
	`hosts: ds1.red.example.com ds2.red.example.com`
	`port: 389`

`-i` *file*	Specifies the PAB LDIF input file.

`-a` *file*	Specifies additional Address Books, one line per book, separated by |. Required values are URL, display name, and description. White space, leading and trailing entries, is ignored. Format:
	
	`Abook: ldap://idir2/c=US|GAL Directory|GAL Directory`
	`ABOOK: ldap://idir3/c=US|Second PAB Directory|Second PAB Directory`

`-l` *file*	The `ugldif` output file, to which both `psroot` and `mepabmigration` are written.
`-o` *file*	The ABS LDIF output file, migrated from the PAB input file. The default is standard output.
`-v`	Verbose option, useful for debugging purposes. This option writes to the `/tmp/debug_log` file.
`-r`	Reverse encoding of special characters encoded by PAB. This is not the default.

How Is the pab2abs.pl Tool Implemented?

The `pab2abs.pl` tool is implemented in Perl (5.8.0), chosen for its rich set of regular expressions, wide availability, and dispensing with separate compilation. Earlier versions might require the MIME::Base64 module, which is available for download from the following location:

`http://cpan.uwinnipeg.ca/module/MIME::Base64`

The `pab2abs.pl` tool works by reading entries in paragraph mode from the input `pab.ldif` file, and matching them with PAB DN, group, or contact as shown in the flow chart in “Appendix A: Logic Flow of pab2abs.pl” on page 11. The PAB DN triggers the creation of address book object classes and sets the value of `PiDefaultAB` to the value of `un` of the `pab` object class. This scheme is able to easily cross-check and immediate identify any untranslated entries. Also, the tool writes `mepabmigration=1` and the value for `psroot` to the `user.ldif` file corresponding to the user. The `PiEntryID` for ABS object classes is based on `time()` in seconds appended with a four-digit hex number, incremented sequentially. Such a string is a valid directory string and conforms to the requirements of address book schema.

Entries that do no match the above requirements are logged as errors.

A group or contact entry is split into lines and each PAB attribute is translated to its corresponding entry or entries as shown in Table 1 on page 4. Embedded special characters (such as $ and line-feed/carriage returns) are encoded. Further, multiline descriptions and notes are encoded in base64 conforming to LDIF RFC 2849. Translation failures are noted in the error log. If necessary, use the `-v` option to generate verbose debugging information. All successfully translated groups and contacts are then counted.

The above procedure is repeated until all entries in the input file are read.

Notes on the Usage of pab2abs.pl

Use the following notes to better understand the behavior of the `pab2abs.pl` tool:

- The `pab2abs.pl` tool is a standalone program and can be run on any system installed with Perl. However, the `ldif2db` program should be run on the back-end server where the ABS resides.
- Errors, including translation failures, and entries skipped and slipped, are written to standard output. You enable logging with the `-v` option, and output is written to the `/tmp/debug_log` file.
- A second mail attribute is migrated to `home` mail address. The `mailAlternateAddress` attribute is not translated.
- The `PiEntryID` attributes for ABS entries are identical to unique IDs (`un`) of the PAB.[1] Unique hex number IDs are generated for other required entries, that is, default address book and corporate address book.
- LDIF requires that any values containing new lines (not a `SAFE-INIT-CHAR`) be base64 encoded. To preserve new-line or line-feed characters contained in PAB, ABS multi line-descriptions must be encoded in base64 using `-r` option.
- The generated `ugldif` (`-l` option) file contains only two entries, for every migrated user: `psroot` and `mepabmigration=1`.
- Upon user login attributes `SunUCPrefs` and `sunAbExtendedUserPrefs: abDefaultView=`*xxx*, will be initialized.
- The tool can accommodate `ou="cn=uid"` instead of `ou=uid`, in the PAB DN.
- The tool accommodates more than one address book. Input to construct these objects is read from a file specified with the `-a` option.
- Parameters from the `-f` option eclipse the `-d`, `-h`, `-b`, and `-p` options.

Best Practices

This section describes best practices to use when migrating the Messenger Express PAB to the Communications Express ABS.

- Before beginning your migration, ensure that your Communications Express deployment is properly configured. For the purpose of address book migration, Communications Express is deemed properly configured if a test user can log in, view, and edit personal contacts successfully. Tacit assumption is that you have completed the installation and configuration procedures from the *Sun Java System Communications Express 6 2005Q4 Administration Guide*.
- Create a few test user accounts for trial runs. Try migrating one of these test accounts by using the `runMigrate.sh` script. You will then have data to compare to accounts that you migrate using the `pab2abs.pl` tool.
- In general, use a migration approach that involves three phases:

1. Migrating a single test user.
2. Migrating a limited number of users (about 50).

1.Address Book identifiers (PiEntryID) are hex numbers generated from current time in milliseconds. An implementation of the `pab2abs.pl` using hex numbers for piEntryID is also available.

3. In the final phase, using logical or physical partitions for dividing the entire PAB, if they exist, and migrating each part separately. An example would be to exploit the natural partition provided by replicas, each serving a distinct set of users. Or, use separate organizational units under `o=pab`. In the latter case, you have to combine the migrated parts before initializing the ABS suffix.

To Migrate Single or Multiple Users

1. Capture all PAB entries of a typical user (a test user) with `ldapsearch`. Include `-1` and `-T` options or `pab2abs.pl` fails.
2. Populate your parameters file based on the template given in “Appendix B: Sample Parameters File” on page 12.
3. Run `pab2abs.pl` to migrate the user's PAB. Name the output file `abs.ldif`. Enable debugging (the `-v` option) during initial trials to quickly identify and correct problems in translation.
4. Modifications to `userRoot`, necessary to prevent further migration attempts, are generated in a file with the -l option. Name the output file `user.ldif`.
5. Using the `ldapmodify` command, load `abs.ldif` and `user.ldif`.
6. Log in as the test user and check the address book to ensure correct migration.

   Resolve schema violations and other inconsistencies that appear in Directory Server log files.

   At this stage, you have successfully migrated a single user.

“Appendix D: Single/Multiple User Migration Pattern” on page 12 contains a pattern that illustrates the routine involved in migrating one or more users. By updating the deployment specific parameters in this script, then using it with `pab2abs.pl`, you have an alternative to `runMigrate.sh` for migrating a few users.

To Migrate an Entire PAB

1. Stop the Directory Server and export `pabdb2` with the `db2ldif` command. Alternately, place the Directory Server in read-only mode, use `db2ldif.pl`. The output LDIF file should be `pab.ldif`.

   Ensure that no unresolved inconsistencies (including those mentioned in this article) are noted in the Directory Server logs.

   Use `db2ldif`, one time, with the required options: `-U`, `-N`, `-u`, `-1`, `-n`, `-s`, and `-a`. You can split the data to carry the transformation of several smaller parts, if so desired (not fully supported yet).

   Example:

   ```
   db2ldif -U1Nu -n pabdb2 -s o=pab -a /tmp/pab.ldif
   ```
2. Export ABS as an LDIF file to preserve the top object classes and any existing address books (ABS). Name the file `abs_gold.ldif`.
3. Run `pab2abs.pl`, input, copy of `pab_gold.ldif`, output, `abs_migrated.ldif`.

   Example:

   `pab2abs.pl -r -f` *params* `-i /tmp/pab.ldif -l /tmp/user.ldif -o /tmp/abs_migrated.ldif`
4. Run the following:

   ```
   cat/tmp/abs_gold.ldif /tmp/abs_migrated.ldif > /tmp/abs_combined.ldif
   ```

The top object classes (`dn: o=PiServerDB and dn: o=`*your-domain*`, o=PiServerDB`) are not generated by the tool. When initializing the suffix, these objects need to be prepended to the `abs.ldif` file.

5. Import/reinitialize the ABS suffix using `ldif2db`, for example:

   ```
   ldif2db -n PiServerDBdb2 -s o=PiServerDb -i /tmp/ab_combined.ldif
   ```

6. Start the Directory Server.
7. Modify user DN to set `mepabmigration=1` and `psroot`.

   When using `ldapmodify`, use the `-c` and `-e` options.

   The migration is now complete.

Note – Posting changes to the user/group subtree to set values of `psroot` and `mePabMigration=1` is the slowest step in the above task. If no dynamic migration were ever to be enabled, the usergroup modification step could be omitted. A default `psroot` will be created upon login. If the default is not satisfactory, then `psroot` value must be set explicitly as desired. You can speed up the process by splitting the file into several parts to be used with different `ldapmodify` command invocations, either on the same Directory Server, or another, or both.

Note – Consult Chapter 8, “Managing Replication,” in the *Sun Java System Directory Server 5.2 2005Q4 Administration Guide*, in planning the overall migration strategy in a complex directory server deployment (cascading/multi-masters). Several methods exist for initializing replicas and hubs. Choose the one that best meets your goals. First, `PiServerDB` is bulk loaded (suffix initialization with `ldif2db`) to one of the masters, and a copy is reexported with the `-r` option. You then work on the other masters, hubs, and replicas, from the reexported copy.

Performance Results from Running the pab2abs.pl Tool

The following data was recorded during the actual migration of a customer PAB database, approximately 2 GBytes in size (about 4 million entries). The exported PAB data was copied to a server, transformed, and copied back for ABS suffix initialization. The following table shows the time, in minutes, of each phase, as well as the total time.

Export db2ldif	Copy	Transformation pab2abs.pl	Copy	Import ldif2db	Total Migration Time
40	10	30	10	120	210

At another customer site, on a PAB approximately 2.5 Gbytes in size, running on a dual-core Opteron server (Solaris™ 10 OS), the transformation took only about 10 minutes.

Future Enhancements to the pab2abs.pl Tool

- Support migration of parts of an exported `pab.ldif` file, particularly when the size of the PAB is greater than 10 GBytes. Note: This enhancement needs to be verified with actual customer data.
- Define a flag for the case where `cn=uid_value` rather than `uid=uid_value` in user DNs.

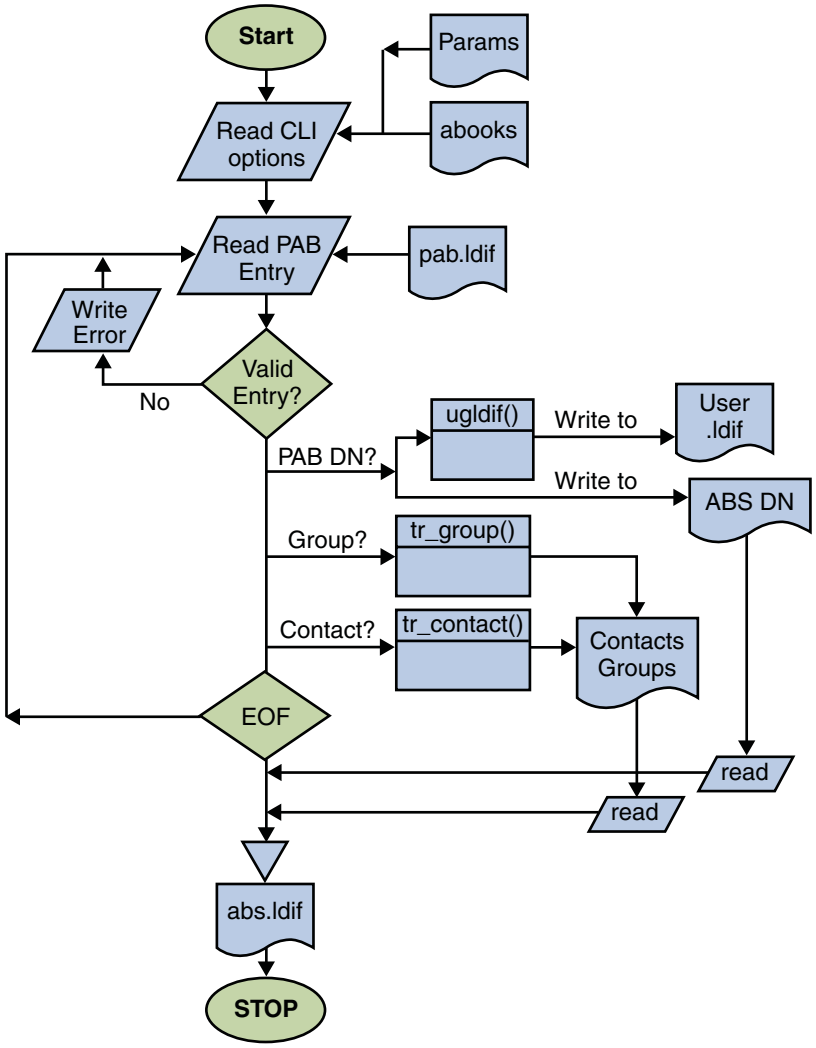
Summary

Migrating from Messenger Express to Communications Express requires that you also migrate the Messenger Express Personal Address Book to the new Communications Express Address Book Store. The `runMigrate.sh` tool bundled with Communicatinos Express has been found to perform slowly with large PAB databases. Consequently, a new tool, `pab2abs.pl` has been released. The `pab2abs.pl` tool uses Sun Java System Directory Server bulk export and import, which is a much faster method compared to creating or modifying entries over LDAP (the method used by the `runMigrate.sh` script). Using `pab2abs.pl` instead of `runMigrate.sh` can reduce your PAB migration times by a factor of thirty or more.

Appendix A: Logic Flow of pab2abs.pl

The following figure shows the logic flow of the `pab2abs.pl` tool.

Figure 2. The pab2abs.pl Tool



Appendix B: Sample Parameters File

The following is a sample parameters file that you use with the `-f` option of the `pab2abs.pl` tool.

```
domain: red.iplanet.com
pabdn: ou=people,o=red.iplanet.com,o=jes3,o=pab
hosts: ds1.red.iplanet.com ds2.red.iplanet.com
port: 389
```

Appendix C: Sample Address Books File

The following is a sample address books file that you use with the `-a` option. Address books, one line per book, are separated by the | character. Required values are: URL, display name and description. White space, leading and trailing the entries are ignored.

```
Abook: ldap://idir2/c=US|HQDA Exchange GAL Directory|This is HQDA Exchange GAL Directory
ABOOK: ldap://idir3/c=US|Leavenworth GAL Directory|This is Leavenworth GAL Directory
```

Appendix D: Single/Multiple User Migration Pattern

```
#!/usr/bin/perl -w
# ------------------------------------------------------------------
#
# Options:
#     d: domain name
#     l: Logfile
#     u: file with list of users
#
# Usage:
#     migrationPattern [ -l logfile ] -d <domain> -u <user_list>
#
# Withot -u options users are read one at a time from STDIN
# Empty input or 'quit' terminates interactive migration.
#
# Set default domain to avoid using -d
# ------------------------------------------------------------------

use Getopt::Std;
use Time::HiRes qw( gettimeofday );

#- ----------------------------------------------------------------
#-  OPTIONS

my %option = ( );

getopts("d:l:u:", %option);
```

(Continued from previous page)

```
my $domain = $option{d} || "red.iplanet.com"  ;
my $pabDN = "ou=people,o=${domain},o=jes3,o=pab" ;

# UserRoot DS Host
my $userRootDS = "hat.red.iplanet.com" ;

# PAB DS Host
my $pabds = "hat.red.iplanet.com" ;

# ABS DS Host
my $absds = "hat.red.iplanet.com" ;

# Password for Directory Manager, Assuming the same password for
# All Directory servers.
my $pw = "nvempati" ;

# uid are under this subtree
my $user_sfx = "ou=people,o=${domain},o=jes3" ;

#: open "-" for input => means STDIN
if ( defined $option{u} ) {
    print "User list from file: $option{u}\n" ;
}
else {
    $option{u} = "-" ;
    print "Enter user name: " ;
    $mode = "tty" ;
}
open(USERS, "< $option{u}")
     or die "Can't open user list: $option{u} for input: $!\n";

#: opening "-" for writing log => means STDOUT
$option{l} = "-" unless defined $option{l};
print "Logfile : $option{l}\n" unless $option{l} eq "-";
open(LOG, "> $option{l}")
     or die "Can't open $option{l} for output: $!\n";

#- ------------------------------------------------------------------
#- MIGRATION

my $count = 0 ;

while (<USERS>) {

    my ($result, $t1, $t2, $userPAB, $userLdif)  ;
    my $uid = $_ ;

    chomp $uid;

    if ( defined $mode ) {
        last if $uid =~ /(quit|^$)/ ;
    }
    next if $uid =~ /^$/ ;
```

(Continued from previous page)

```
    $count ++ ;
    $userPAB = "ou=$uid,$pabDN" ;

    $t1 = gettimeofday();

    # 1. Capture PAB in a LDIF file
    $result = system "ldapsearch -h $pabds -T1D 'cn=directory manager' -w $pw -b $userPAB
objectclass=* > /tmp/$uid.pab.ldif";
    if ( ($result / 256) > 0 ) {
        print "LDAP_ERROR: $result\n" ;
    }

    # 2. Convert PAB to ABS
    $result = system "pab2abs.pl -b $pabDN -d $domain -h $absds -i /tmp/${uid}.pab.ldif -o /tmp/
${uid}.abs.ldif";
    if ( ($result / 256) > 0 ) {
        print "LDAP_ERROR: $result\n" ;
    }

    # 3. Load the ABS LDIF to PiServerDB
    $result = system "ldapmodify -h $absds -D 'cn=directory manager' -w $pw -q -c -a -f /tmp/
$uid.abs.ldif";
    if ( ($result / 256) > 0 ) {
        print "LDAP_ERROR: $result\n" ;
    }

    # 4. UserRoot modifications to prevent further migration
    open (USER_LDIF, ">/tmp/${uid}.user.ldif") || die "cant write user ldif file\n";
    ( $userLdif = << "____UserRootChanges" ) =~ s/^\s+//gm ;
        dn: uid=${uid},${user_sfx}
        changetype: modify
        add: psroot
        psroot: ldap://${absds}:389/piPStoreOwner=${uid},o=${domain},o=PiServerDb
        -
        add: nswmExtendedUserPrefs
        nswmExtendedUserPrefs: mepabmigration=1

____UserRootChanges

    print USER_LDIF  "$userLdif" ;
    close (USER_LDIF);

    # 5. Load the userRoot changes
    $result = system "ldapmodify -h $userRootDS -D 'cn=directory manager' -w $pw -q -c -a -f /tmp/
${uid}.user.ldif";
    if ( ($result / 256) > 0 ) {
        print "LDAP_ERROR: $result\n" ;
    }
$t2 = gettimeofday();

    printf LOG "Migration of %s took %.4f seconds\n", $uid, $t2 - $t1 ;

if ( defined $mode ) {
        print "Enter user name: " ;
```

(Continued from previous page)

```
    }
}

print LOG "Migrated $count users\n" ;

1;
```

About the Author

Sarma Vempati has worked at Sun Microsystems for the past 8 years. Previously he was a consultant at Hewlett Packard, Silicon Graphics, and Auspex. His other appointments include working at Health Information Science, University of Victoria and British Columbia Systems, Victoria, BC. He holds three graduate degrees including one from Indian Institute of Technology, Kanpur, India.

Acknowledgements

The author would like to recognize the following individuals for their contributions to this article:

- Sun engagement managers Gary Cooper and Bob Caine for their generous support and for the opportunity to perfect the tool with US Army PAB data.
- Sun engineers Zein Shivji, Paul Welcome, Martin Thiele, and Troy Lowe for their valuable suggestions.
- Allyn Polk, for his constant support and encouragement, Chuck Alexander for seeding the BluePrints idea, and Jonathan Hawkins for connecting me with the right people.
- Sun technical writer Joe Sciallo, for his editorial support.
- Anil Kowshik, Jason Walters, and Craig Binny for their interest and support, and for providing the opportunity to work with Shaw Cable, Calgary, Canada.

References

- Good, G. “The LDAP Data Interchange Format (LDIF) - Technical Specification, RFC 2849,” June 2000
- Chapin, N. “Flowcharting with the ANSI Standard: A Tutorial, Computing Surveys, Vol 2,” June 1970
- Chapter 2, “Installing and Configuring Communications Express,” *Sun Java System Communications Express 6 2005Q4 Administration Guide*
- *Sun Java System Directory Server 5.2 2005Q4 Administration Guide*

Related Resources

Download the `pab2abs.pl` script at:

- `http://www.sun.com/blueprints/tools/`

Information on Sun Java System Messaging Server, Sun Java System Messenger Express, and Sun Java System Communications Express is available at:

- `http://www.sun.com/bigadmin/hubs/comms`

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`