```
#!/usr/bin/perl -I/export/home/axnsoc/bin

##########################################################################
# batchadd.pl <file> => queued processing of account creations          #
##########################################################################
# Order of operation
# Connect to RPTP
# execute query, capture results
# Bind to Directories
# Read Record, Parse SSN
# If 30mill exists in Prod, update it, log username changes for callcenter.
# If 30mill does not exist, create a new account.
# Repeat through Records
# Unbind Directories


use uaaldap;
use Net::LDAP qw(:all);
use Net::LDAP::Util qw(ldap_error_name ldap_error_text);
use DBI;
use Data::Dumper;

$DBI_CONN_STRING = "dbi:Oracle:****"
$DB_USER = "***";
$DB_PASSWORD = "***";

$LOGFILE = "batch_all_admitted_students.log";
$LOGPATH = "/export/home/axnsoc/data/";
$LOG = $LOGPATH . $LOGFILE;
$LOG = "/dev/null"; # jim switch to stdout logging and uses bash to pipe output to logs in crontab

$ENV{ORACLE_HOME} = '/u2/oracle/';

# Production Server - server has since been decommissioned
$pserver = "sentry.uaa.alaska.edu:389";

# Archive Server - server has since been decommissioned
#$aserver = "sentry.uaa.alaska.edu:4389";

$base = "o=uaa.alaska.edu,o=isp";
$basedn = "ou=student,o=uaa.alaska.edu,o=isp";
$abasedn = "ou=admin,o=uaa.alaska.edu,o=isp";

# Directory Creditials
$admindn = "***";
$passwd = "***";


# Default Schema Settings
@objectclass        = ( "top", "person", "organizationalPerson", "inetorgperson", "uaainetorgperson",
                "inetUser","ipUser","nsManagedPerson","userPresenceProfile","inetMailUser",
                "inetLocalMailRecipient");
$maildomain         = "uaa.alaska.edu";
$mailhost           = "mail02.uaa.alaska.edu";
$mail               = "undefined";
$maildeliveryoption = "forward";
$mailmessagestore   = "stu";
$mailquota      = 65536;
$activated          = 1;
$dusage             = 0;

$QUERY = "SELECT DISTINCT stvcamp_code affil_camp_code, spriden_pidm Pidm, spriden_id UAID, gobtpac_external_user Global_ID,
        spriden_last_name||\', \'||spriden_first_name||\' \'||spriden_mi NAME,
        spriden_first_name First_name, spriden_mi Middle_name, spriden_last_name Last_name,
        SUBSTR(spriden_first_name,1,1)||SUBSTR(spriden_mi,1,1)||SUBSTR(spriden_last_name,1,1)||\'!\'||SUBSTR(spriden_id,5,4) Default_Password
```

```
FROM spriden, sfrstcr, gobtpac, stvcamp, stvrsts
WHERE spriden_pidm = sfrstcr_pidm
AND spriden_pidm = gobtpac_pidm
AND spriden_change_ind IS NULL
AND spriden_id NOT LIKE \’BAD%\’
AND sfrstcr_credit_hr >= 0
AND stvrsts_code = sfrstcr_rsts_code
AND stvrsts_incl_sect_enrl = \’Y\’
AND sfrstcr_camp_code IN (\’A\’,\’B\’,\’D\’,\’I\’,\’P\’,\’V\’)
AND stvcamp_code = sfrstcr_camp_code
AND sfrstcr_term_code >= (SELECT MAX(stvterm_code) FROM stvterm
                    WHERE SYSDATE BETWEEN stvterm_start_date AND stvterm_end_date
                      AND SUBSTR(stvterm_code,5,2) IN (\’01\’,\’02\’,\’03\’) )
UNION
SELECT DISTINCT application_campus_code affil_camp_code,
spriden_pidm Pidm, spriden_id UAID, gobtpac_external_user Global_ID,
spriden_last_name||\’, \’||spriden_first_name||\’ \’||spriden_mi NAME,
spriden_first_name First_name, spriden_mi Middle_name, spriden_last_name Last_name,
SUBSTR(spriden_first_name,1,1)||SUBSTR(spriden_mi,1,1)||SUBSTR(spriden_last_name,1,1)||\’!\’||SUBSTR(spriden_id,5,4) Default_Password
FROM admissions_basic_view, spriden, gobtpac
WHERE application_campus_code IN (\’A\’,\’B\’,\’D\’,\’I\’,\’P\’,\’V\’)
AND application_status = \’D\’
AND decision_code IN (\’AB\’,\’AC\’,\’AD\’,\’AO\’,\’AP\’,\’AX\’,\’AZ\’,\’SB\’,\’SC\’,\’SD\’,\’SO\’,\’SP\’,\’SX\’,\’SZ\’)AND application_term >= (SELECT MAX(stvterm_code) FROM stvterm
                    WHERE SYSDATE BETWEEN stvterm_start_date AND stvterm_end_date
                      AND SUBSTR(stvterm_code,5,2) IN (\’01\’,\’02\’,\’03\’) )
AND student_id = spriden_id
AND spriden_change_ind IS NULL
AND spriden_pidm = gobtpac_pidm
AND spriden_id NOT LIKE \’BAD%\’
";

$count = 0;

%expiration_date = (    "1" => "0924",
            "2" => "0924",
            "3" => "0924",
            "4" => "0924",
            "5" => "0924",
            "6" => "0924",
            "7" => "0124",
            "8" => "0124",
            "9" => "0124",
            "10" => "0124",
            "11" => "0124",
            "12" => "0924" );

%year_map   = (    "1" => "0",
                        "2" => "0",
                        "3" => "0",
                        "4" => "0",
                        "5" => "0",
                        "6" => "0",
                        "7" => "1",
                        "8" => "1",
                        "9" => "1",
                        "10" => "1",
                        "11" => "1",
                        "12" => "1" );

$edate = &get_edate();

# Ok, now let’s get to work.

$START_TIME = &get_timestamp();

# open the log file so we can save our efforts
open(LH,">>$LOG") || die("Error, can not open file: ", $LOG," due to: ", $!,"\n");
```

```perl

#&log(LH, "####################################################################################\n");
#&log(LH, "Starting synchronization process...\n");
#&log(LH, "Opening database connection...\n");

# Open DB connection
$dbh = DBI->connect($DBI_CONN_STRING, $DB_USER, $DB_PASSWORD) || die();

#&log(LH, "Binding to Directory Servers...\n");

my $pldap = Net::LDAP->new("$pserver") or die "$@";     # Bind to the Production Directory

my $pmsg = $pldap->bind(dn=>"$admindn", password => "$passwd", version => 3);
if ($pmsg->code){ # If code != 0 LDAP_SUCCESS then we complain
    die("Error: Unable to connect to $pserver: $@\n");
}

#my $aldap = Net::LDAP->new("$aserver") or die "$@";     # Bind to the Archive Directory

#my $amsg = $aldap->bind(dn=>"$admindn", password => "$passwd", version => 3);
#if ($amsg->code){ # If code != 0 LDAP_SUCCESS then we complain
#        die("Error: Unable to connect to $aserver: $@\n");
#}

# ok, all services are open and connected at this point.

$query = $dbh->prepare($QUERY);

#&log(LH, "Running query...\n");
$query->execute();

#&log(LH, "Query completed...\n");
#&log(LH, "Processing LDAP entries with queried data...\n");

# Tom Riley used integers to identify the columns from the sql query. This is the integer map to understand the next block
# 0 affil_camp_code
# 1 Pidm
# 2 UAID
# 3 Global_ID
# 4 NAME
# 5 First_name
# 6 Middle_name
# 7 Last_name
# 8 Default_Password
while(@row = $query->fetchrow_array()) {


    # get colums from the array into variables
    $loc=$row[0];
    $ssn=$row[2];
    $globalid=$row[3];
    $fullname=$row[4];
    $givenname=$row[5];
    $mi=$row[6];
    $lastname=$row[7];
    $passwd=$row[8];

    # scrub double spaces
    $fullname =~ s/   / /g;
    $fullname =~ s/  / /g;

    $givenname =~ s/   / /g;
    $givenname =~ s/  / /g;

    $mi =~ s/   / /g;
    $mi =~ s/  / /g;

    $lastname =~ s/   / /g;
    $lastname =~ s/  / /g;


    # scrub middle initial of space, underscore, period
    $mi =~ s/\_//g;
    $mi =~ s/ //g;
    $mi =~ s/\.//g;

    # build initials
    $initials = substr($givenname,0,1) . substr($mi,0,1) . substr($lastname, 0,1);

    # scrub initials of underscore, space
    $initials =~ s/\_//g;
    $initials =~ s/' '//g;  # explicitedly remove spaces from initials.

    # We don't create B type accounts
    if ($loc eq "B") {
        $loc = "A";
    }

    # make AS---#
    $type = $loc . "S"; # Since all these are for student processing

    #   print("\t\tDEBUG: type: $type   initials: $initials\n");

    if ($ssn ne "") { # If it's a valid record

        # search for the user in the student OU - need to look for xs in front of the 30s for gmail accounts
        $studentQuery = $pldap->search(base=>$basedn, filter=>"(|(uniqueidentifier=$ssn)(uniqueidentifier=x$ssn))");

        # search for the user in the Admin OU this occurs since new employees also use the UA username standard
        $adminQuery = $pldap->search(base=>$abasedn, filter=>"(|(uniqueidentifier=$ssn)(uniqueidentifier=x$ssn))");

        #&log(LH, "Searching for existence of \"$ssn $givenname $mi $lastname\"...\n");

        # if the user exists in the student OU then update the identity w/ banner data
        if ($studentQuery->count) {
            # SSN Match, Account exists in Prod
            #&log(LH, "Found.\n");
            my @entries = $studentQuery->entries;
            for $entry (@entries) {
                my $dn = $entry->dn();

                #&log(LH, "update|dn=($dn)|");


                my $logOp = "update";
                my $logData = "dn=$dn|uniqueIdentifier=$ssn|cn=$fullname|givenname=$givenname|middlename=$mi|sn=$lastname|mail=$mail|globalid=$globalid|password=notShown";
                my $logResult = "none";
                my $logResultMsg = "updates disabled";


                my ($u_code, $update_error_msg) = &update_info($pldap, $entry, $givenname, $lastname, $mi, $globalid );  # Update Name information to include UA username


                if ($u_code != 0) {
                    $logResult = "error";
                    $logResultMsg = $update_error_msg;
                }
                else {
                    $logResult = "success";
                    $logResultMsg = $update_error_msg;
                }
                &log(LH, "$logOp|$logResult|$logResultMsg|$logData\n");

            }
        }
        elsif ($adminQuery->count)      {
                #user exists in Admin OU do nothing except maybe log
        }
        else { # No existing records were found, create a new account.
            # First, build all the required pieces.
            $uid = $globalid; #&find_unique_uid($type, $initials);
            $dn = "uid=$uid,ou=student,o=uaa.alaska.edu,o=isp";
            $cn = "$givenname $mi $lastname";
            $fullname = "$givenname $mi $lastname";
            $mail = "$uid\@$maildomain";

            my $password = "Uaa!" . substr($ssn,4,4);

            # scrub double spaces from CN
            $cn =~ s/   / /g;
            $cn =~ s/  / /g;

            my $logOp = "create";
            my $logData = "dn=$dn|uniqueIdentifier=x$ssn|cn=$cn|givenname=$givenname|middlename=$mi|sn
=$lastname|mail=$mail|globalid=$globalid|password=$password";
            my $logResult = "debug";
            my $logResultMsg = "debug message";

           $addmsg = $pldap->add(dn => "$dn",
                      attr=> ['cn' => "$cn",
                                  'sn' => "$lastname",
                                          'givenname' => "$givenname",
                                          'middlename' => "$mi",
                              'objectclass' => [@objectclass],
                              'mail' => "$mail",
                                  'mailhost' => "$mailhost",
                            'mailmessagestore' => "$mailmessagestore",
                            'maildeliveryoption' => "$maildeliveryoption",
                            'mailforwardingaddress' => $globalid . '@alaska.edu',
                            'mailquota' => "$mailquota",
                            'uid' => "$uid",
                            'uniqueidentifier' => "x$ssn",
                            'userpassword' => "$password",
                            'activated'     => "$activated",
                              'fullname' => "$fullname",
                              'globalid' => "$globalid",
                              'expirationdate' => "$edate",
                            'mailallowedserviceaccess' => "+all:\*",
                            'mailuserstatus' => "active",
                            'dataSource' => "NDA 4.5 Delegated Administrator",
                            'inetUserStatus' => "active"]
                       );
                $add_code = $addmsg->code;
                $add_error_msg = ldap_error_name($add_code);

                if ($add_code != 0) {
                    # LDAP_SUCESS = 0, so this is an error
                    $logResult = "error";
                    $logResultMsg = $add_error_msg;
                    #&log(LH, "Error",$add_error_msg,"\n");
                }
                else {
                    $logResult = "success";
                    $logResultMsg = "success";
                    #&log(LH, "Success\n");
                }
                &log(LH, "$logOp|$logResult|$logResultMsg|$logData\n");

        }
    }
}
$pldap->unbind;
#$aldap->unbind;


sub update_info {

    # incorporated Jim Weller's changes from original script, modified by Joe to include globalid handling (UA username)

        # the ldap connection
        my $ldap = shift;

        # the ldap entry of a single user account to be checked for name correctness
        my $entry = shift;

        # the atomic name values
        my $newGn = shift; # first name
        my $newSn = shift; # last name
        my $newMi = shift; # middle name
        my $newGlobalid = shift; # ua username

        # this returns the exact location in ldap of this user account
        my $dn = $entry->dn();

        # a tracking bit, if one of the first, middle or last names changes, this will be set to 1
        my $somethingChanged = 0;
        #tracking bit for uaUsername
        my $uaChanged = 0;

        # a trivial search in $dn to get the person's name values
        my $qs =  $ldap->search(base=>$dn, filter=>"(objectclass=*)", attrs=>["fullname","sn","givenname","middlename","cn","globalid"]);

        my $tmpMsg = '';

        my $u='';
        my $u_code='';
        my $u_msg='';
        # if the search is successful (which it should be because it is the
        #  result of a successful search before this function is called
        if ($qs->count() > 0) {
                my $t_entry = $qs->entry(0);

                $currentGn = &get_attr($t_entry, "givenname");
                $currentMi = &get_attr($t_entry, "middlename");
                $currentSn = &get_attr($t_entry, "sn");
                $currentGlobalid = &get_attr($t_entry, "globalid");

                # check if the first name has changed
                if( $currentGn ne $newGn )
                {
                        $somethingChanged = 1;
                }

                # check if the middle name has changed
                if( $currentMi ne $newMi )
                {
                        $somethingChanged = 1;
                }

                # check if the last name has changed
                if( $currentSn ne $newSn )
                {
                        $somethingChanged = 1;
                }

                #check if the ua username changed (ie married, divorced, etc.)
                if ( $currentGlobalid ne $newGlobalid)
                {
                        $uaChanged = 1;
                }
                                                                #globalid changed, manual modification required log and move on
                                                                if($uaChanged == 1) {
                                                                                $u_msg = "rename required: ($currentGlobalid,$newGlobalid)";
                                                                                $u_code = 0;
                                                                }
                #one of the atomic base names changed, update fullname and cn
                elsif( $somethingChanged == 1 )
                {
                        $tmpMsg =  "givenname($currentGn ,$newGn),";
                        $tmpMsg .= "middlename($currentMi ,$newMi),";
                        $tmpMsg .= "sn($currentSn ,$newSn),";

                        $u = $ldap->modify($dn, replace => {'givenname' => $newGn,
                                                            'middlename' => $newMi,
                                                            'sn' => $newSn,
                                                            'cn' => $fullname,
                                                            'fullname' => $fullname}
                                                            );

                        $u_msg = ldap_error_name($u->code) . ":" . $tmpMsg;
                 }
                 else
                 {
                         $u_code = 0;
                         $u_msg = "no update necessary";
                 }
        }
        return ($u_code,$u_msg);


}

sub update_mailquota {

        my $ldap = shift;
        my $entry = shift;

        my $dn = $entry->dn();
        my $u = $ldap->modify($dn, replace => { 'mailquota' => $mailquota});

        return $u;

}

sub copy_to_ds {
        my $tldap = shift;
        my $entry = shift;
        my $res = $tldap->add($entry);
        return $res;
}

sub remove_from_ds {
        my $ldap = shift;
        my $entry = shift;
        my $res = $ldap->delete($entry);
        return $res;
```

```
}

sub set_archive_bit {
        my $ldap = shift;
        my $entry = shift;
        my $bit = shift;
        my $res = $ldap->modify($entry->dn(), replace=>{'activated'=>$bit});
        return $res;
}

#this routine was used back in the pre-UAusername days and is only left in for refer
ence, never call this
sub find_unique_uid {
    # Both, pldap and aldap are defined and useable
    my $t = shift;
    my $i = shift;
    my $uid = join("",$t,$i);

    #ok, UID is built, start querying

    &log(LH, "\t\tGenerating Userid...\n");

    $prodtq = $pldap->search(base=>$basedn, filter=>"(uid=$uid)"); # Production Query

    $archtq = $aldap->search(base=>$basedn, filter=>"(uid=$uid)"); # Archive Query

    if ($prodtq->count == 0 && $archtq->count == 0) {
        # The supplied account is valid;
        &log(LH, "$uid available.\n");
    }
    else {
        &log(LH, "$uid not available.\n");
        my $found = 0;
        my $idx = 1;
        while(!$found) {
            my $id = join("",$uid,$idx);
            &log(LH, "\t\tChecking \"$id\"...");
            my $pchk = $pldap->search(base => $basedn, filter=>"(uid=$id)");
            my $achk = $aldap->search(base => $basedn, filter=>"(uid=$id)");
            if ($pchk->count || $achk->count) {
                # Match Found, not available, try the next number
                $idx++;
                &log(LH, "Not available.\n");
            }
            else {
                $uid = $id;
                $found = 1;
                &log(LH, "Available!\n");
            }
        }
    }
    $uid = lc($uid);
    return $uid;

}

sub get_edate {
    my ($mon, $year) = (localtime)[4..5];
#   print("DEBUG: mon $mon   year: $year\n");
    my $year = $year + 1900 + $year_map{$mon};
    my $edate = $expiration_date{$mon};
    my $ed = "$year$edate";
    return $ed;
}

sub log {

    local(*LH) = @_[0];
        my $msg = @_[1];
        print($msg);
}

sub get_timestamp {
        my ($s, $m, $h, $day, $mon, $year, $wd, $yday, $isd) = localtime();
        $mon = $mon+1;
        my $datestr = $year+1900 . ":$mon:$day:$h:$m:$s";
        return $datestr;
}
```