System Administration: Patron SIF Utility Script

Created By: Matthew Hooper
Created on: 5/15/2019

[This article was copied from the Voyager Wiki.]

Utility script for voyager patron SIF record format. This script allows the user to create a blank patron SIF record, read a SIF record from string input, write a SIF record to file output. The records are created as data objects so individual elements of a SIF record can be accessed without having to calculate string offsets. Useful for those who have to automate the loading of patron records and just want the abstract.

```
#!/m1/shared/bin/perl


# Standard patron record include file for loading patron records with correct data structure.

# Includes functions to create a blank patron record, read a record from string input, and print a record to file output


# based on voyager technical manual definition of SIF file format for voyager unicode - should work for any 2001+ version


# Patron SIF perl script v 1.01 - 18/04/2005

# Author Matt Hooper @ Flinders University - Matthew.Hooper@flinders.edu.au

# No guarantees that this include file will work as advertised - test it first!!


######################################## HOW TO USE THIS FILE
############################################################


# 1) Create your own perl script with the statement

#          require '$PATH_TO_THIS_FILE/patron_sif_defaults.pl' ;
```

```
# 2) Change any defaults in the procedure new_sif_record

# 3) Call the functions as follows:

#         $record = &new_sif_record($n); - creates a new blank patron sif record with n
addresses

#         $record = &read_sif_record($string); - reads a sif record from string input

#         &write_sif_record($record,"FILEHANDLE"); - writes sif record to file output
given filehandle

# 4) Since I've only written the 3 functions you can pull out SIF record field values
from a record by doing something

#    like: $inst_id = $record -> {'inst_id'};

#      or: $patron_address2_line1 = $record -> {'address_line12'};

# more functions can be added on request. I'm hoping to add some that take away the need
to remember pack()

# function syntax and substr()


################################################################################################


# date functions - sets current date in SIF format

($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime;

 $year = $year + 1900;

 $mon = $mon + 1;

 if ($mon < 10){$mon = "0".$mon;}

 if ($mday < 10){$mday = "0".$mday;}

 # need today's date

 $today = $year."-".$mon."-".$mday;


# function to create a blank patron record with X address segments

# returns a pointer to a hash so the result can be referenced between functions/
procedures
```

```
sub new_sif_record()

{

# some defaults for the functions => change these if the default value for the created
records needs to be changed

my $default_patron_expiry_date = ($year+5).".03.31";

# this should be a valid location code

my $default_load_location = "XRG";

my $default_patron_purge_date = ($year+10).".03.31";


 my $address_count = @_[0];

 # this is the patron record - hash of field values

 my %temp = {};

 # this is a pointer to the hash

 my $record_pointer = 0;

 #fill record details ready for writing SIF stub

 $temp{'patron_id'} = pack("A10",' ');

 # write barcode fields

 for $t(1..3)

 {

   $temp{"patron_barcode_id".$t} = pack("A10",' ');

   $temp{"patron_barcode".$t} = pack("A25",' ');

   $temp{"patron_group".$t} = pack("A10",' ');

   $temp{"patron_barcode_status".$t} = pack("A1",' ');

   $temp{"patron_barcode_modified_date".$t} = pack("A10",' ');

 }


 # set registration date to be load date by default

 $temp{'registration_date'} = $today ;
```

```
$temp{'patron_expiration_date'} = $default_patron_expiry_date ;

$temp{'patron_purge_date'} = $default_patron_purge_date ;

$temp{'voyager_date'} = pack("A10",' ');

$temp{'voyager_updated'} = pack("A10",' ');

$temp{'library_location_code'} = pack("A10",$default_load_location);

$temp{'inst_id'} = pack("A30",' ');

$temp{'ssn'} = pack("A11",' ');

# initialize stat codes - all ten of them

for $t(1..10)

   {$temp{"stat_category".$t} = pack("A3",' ');}

# patron name data - default to personal

$temp{'name_type'} = 1;

$temp{'surname'} = pack("A30",' ');

$temp{'first_name'} = pack("A20",' ');

$temp{'middle_name'} = pack("A20",' ');

$temp{'title'} = pack("A10",' ');

# now for transaction counters. Since default should be nothing

# and we would probably never want to load new records with these

# set to anything else, then just load them as a string of blanks

$temp{'transaction_counters'} = pack("A65",' ');

$temp{'address_count'} = $address_count;


for $t(1..$temp{'address_count'})

{

   $temp{"address_id".$t} = pack("A10",' ');

   $temp{"address_type".$t} = ' ';

   $temp{"address_status_code".$t} = "n";

   $temp{"address_begin_date".$t} = pack("A10", ' ');
```

```
    $temp{"address_end_date".$t} = pack("A10", ' ');
    $temp{"address_line1".$t} = pack("A50",' ');
    $temp{"address_line2".$t} = pack("A40",' ');
    $temp{"address_line3".$t} = pack("A40",' ');
    $temp{"address_line4".$t} = pack("A40",' ');
    $temp{"address_line5".$t} = pack("A40",' ');
    $temp{"city".$t} = pack("A40",' ');
    $temp{"state".$t} = pack("A7",' ');
    $temp{"zip_post".$t} = pack("A10",' ');
    $temp{"country".$t} = pack("A20",' ');
    $temp{"phone_primary".$t} = pack("A25",' ');
    $temp{"phone_mobile".$t} = pack("A25",' ');
    $temp{"phone_fax".$t} = pack("A25",' ');
    $temp{"phone_other".$t} = pack("A25",' ');
    $temp{"date_add_update".$t} = pack("A10",' ');
 }
 $record_pointer = \%temp;
 # now return the pointer.
 return $record_pointer ;
}

# now some functions to read/write SIF records

# this function takes in a string and returns a hash reference with all the details
created by function new_patron_record
# overlayed with data supplied in the string.
sub read_sif_record()
{
```

```
  my $in_string = @_[0];

  my $record_pointer = 0;

  # $pos_pointer marks where in the input string we are up to, also used to alert to
where in string errors occurred

  my $pos_pointer = 0;

  # do some consistency checking

  # length of string must be at least 1 base segment and 1 address ie 456 +429
characters long (+ EOL marker)

  if (length($in_str) < 885 )

  {

    # record does not conform to base SIF record length

    print "Record does not conform to rules on base SIF record length. Returning null
pointer...\n";

    return 0;

  }

  # otherwise procede with address count

  my $address_count = substr($in_string,455,1);

  if (($address_count < 1) or ($address_count > 9))

  {

    # address count is not correct or wrong data type ie record string has address count
at wrong offset

    print "Address count does not conform to rules on base SIF record. Returning null
pointer...\n";

    return 0;

  }

  # create a new patron record with correct address count to use as a template

  $record_pointer = &new_patron_record($address_count);

  # now populate record with data from input string

  $record_pointer -> {'patron_id'} = substr($in_string,$pos_pointer, 10);
```

```
  $pos_pointer = $pos_pointer + 10;

   for $t(1..3)

  {

   $record_pointer -> {'patron_barcode_id'.$t} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

   $record_pointer -> {'patron_barcode'.$t} =
substr($in_string,$pos_pointer,25);$pos_pointer = $pos_pointer + 25;

   $record_pointer -> {'patron_group'.$t} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

   $record_pointer -> {'patron_barcode_status'.$t} =
substr($in_string,$pos_pointer,1);$pos_pointer = $pos_pointer + 1;

   $record_pointer -> {'patron_barcode_modified_date'.$t} =
substr($in_string,$pos_pointer,10); $pos_pointer = $pos_pointer + 10;

  }

  $record_pointer -> {'registration_date'} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

  $record_pointer -> {'patron_expiration_date'} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

  $record_pointer -> {'patron_purge_date'} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

  $record_pointer -> {'voyager_date'} = substr($in_string,$pos_pointer,10);$pos_pointer
= $pos_pointer + 10;

  $record_pointer -> {'voyager_updated'} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

  $record_pointer -> {'library_location_code'} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

  $record_pointer -> {'inst_id'} = substr($in_string,$pos_pointer,30);$pos_pointer =
$pos_pointer + 30;

  $record_pointer -> {'ssn'} = substr($in_string,$pos_pointer,11);$pos_pointer =
$pos_pointer + 11;


 # initialize stat codes - all ten of them

 for $t(1..10)
```

```
    {$record_pointer -> {'stat_category'.$t} =
substr($in_string,$pos_pointer,3);$pos_pointer = $pos_pointer + 3;}


 # patron name data

 $record_pointer -> {'name_type'} = substr($in_string,$pos_pointer,1);$pos_pointer =
$pos_pointer + 1;

 $record_pointer -> {'surname'} = substr($in_string,$pos_pointer,30);$pos_pointer =
$pos_pointer + 30;

 $record_pointer -> {'first_name'} = substr($in_string,$pos_pointer,20);$pos_pointer =
$pos_pointer + 20;

 $record_pointer -> {'middle_name'} = substr($in_string,$pos_pointer,20);$pos_pointer =
$pos_pointer + 20;

 $record_pointer -> {'title'} = substr($in_string,$pos_pointer,10);$pos_pointer =
$pos_pointer + 10;


 # now for transaction counters.

 $record_pointer -> {'transaction_counters'} =
substr($in_string,$pos_pointer,65);$pos_pointer = $pos_pointer + 65;

 # address count already correct so no need to initialize


 # add address fields - check first one to make sure that it's the permanet address and
all the others should temp or email

 for $t(1..($record_pointer -> {'address_count'}))

 {

    $record_pointer -> {'address_id'.$t} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

    $record_pointer -> {'address_type'.$t} = substr($in_string,$pos_pointer,1);

    if ((substr($in_string,$pos_pointer,1) ne "1") and ($t == 1) )

    {print "Address type at position $pos_counter must be a permanent address. Returning
null pointer...\n"; return 0;}

    elsif((substr($in_string,$pos_pointer,1) eq "1") and ($t != 1) )
```

```
    {print "Address type at position $pos_counter cannot be a permanent address.
Returning null pointer...\n"; return 0;}

    $pos_pointer = $pos_pointer + 1;

    $record_pointer -> {'address_status_code'.$t} =
substr($in_string,$pos_pointer,1);$pos_pointer = $pos_pointer + 1;

    $record_pointer -> {'address_begin_date'.$t} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

    $record_pointer -> {'address_end_date'.$t} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

    $record_pointer -> {'address_line1'.$t} =
substr($in_string,$pos_pointer,50);$pos_pointer = $pos_pointer + 50;

    $record_pointer -> {'address_line2'.$t} =
substr($in_string,$pos_pointer,40);$pos_pointer = $pos_pointer + 40;

    $record_pointer -> {'address_line3'.$t} =
substr($in_string,$pos_pointer,40);$pos_pointer = $pos_pointer + 40;

    $record_pointer -> {'address_line4'.$t} =
substr($in_string,$pos_pointer,40);$pos_pointer = $pos_pointer + 40;

    $record_pointer -> {'address_line5'.$t} =
substr($in_string,$pos_pointer,40);$pos_pointer = $pos_pointer + 40;

    $record_pointer -> {'city'.$t} = substr($in_string,$pos_pointer,40);$pos_pointer =
$pos_pointer + 40;

    $record_pointer -> {'state'.$t} = substr($in_string,$pos_pointer,7);$pos_pointer =
$pos_pointer + 7;

    $record_pointer -> {'zip_post'.$t} = substr($in_string,$pos_pointer,10);$pos_pointer
= $pos_pointer + 10;

    $record_pointer -> {'country'.$t} = substr($in_string,$pos_pointer,20);$pos_pointer
= $pos_pointer + 20;

    $record_pointer -> {'phone_primary'.$t} =
substr($in_string,$pos_pointer,25);$pos_pointer = $pos_pointer + 25;

    $record_pointer -> {'phone_mobile'.$t} =
substr($in_string,$pos_pointer,25);$pos_pointer = $pos_pointer + 25;

    $record_pointer -> {'phone_fax'.$t} =
substr($in_string,$pos_pointer,25);$pos_pointer = $pos_pointer + 25;

    $record_pointer -> {'phone_other'.$t} =
```

```
substr($in_string,$pos_pointer,25);$pos_pointer = $pos_pointer + 25;

    $record_pointer -> {'date_add_update'.$t} =
substr($in_string,$pos_pointer,10);$pos_pointer = $pos_pointer + 10;

 }

  return $record_pointer ;

}


# writes a SIF record to output using hash reference and filehandle as argument

# ie write_sif($test, "THIS_FILEHANDLE"); where THIS_FILEHANDLE is the result of a open
file statement

sub write_sif_record()

{

  my $record_pointer = @_[0];

  my $filehandle = @_[1];

   print $filehandle $record_pointer -> {'patron_id'};

   for $t(1..3)

  {

   print $filehandle $record_pointer -> {'patron_barcode_id'.$t} ;

   print $filehandle $record_pointer -> {'patron_barcode'.$t} ;

   print $filehandle $record_pointer -> {'patron_group'.$t} ;

   print $filehandle $record_pointer -> {'patron_barcode_status'.$t} ;

   print $filehandle $record_pointer -> {'patron_barcode_modified_date'.$t} ;

  }

  print $filehandle $record_pointer -> {'registration_date'};

  print $filehandle $record_pointer -> {'patron_expiration_date'} ;

  print $filehandle $record_pointer -> {'patron_purge_date'} ;

  print $filehandle $record_pointer -> {'voyager_date'} ;

  print $filehandle $record_pointer -> {'voyager_updated'} ;
```

```
 print $filehandle $record_pointer -> {'library_location_code'} ;
 print $filehandle $record_pointer -> {'inst_id'} ;
 print $filehandle $record_pointer -> {'ssn'} ;

# initialize stat codes - all ten of them
for $t(1..10)
   {print $filehandle $record_pointer -> {'stat_category'.$t} ;}

# patron name data
print $filehandle $record_pointer -> {'name_type'} ;
print $filehandle $record_pointer -> {'surname'} ;
print $filehandle $record_pointer -> {'first_name'} ;
print $filehandle $record_pointer -> {'middle_name'} ;
print $filehandle $record_pointer -> {'title'} ;

# now for transaction counters.
print $filehandle $record_pointer -> {'transaction_counters'} ;
print $filehandle $record_pointer -> {'address_count'} ;

# add address fields - check first one to make sure that it's the permanet address and
all the others should temp or email
for $t(1..($record_pointer -> {'address_count'}))
{
   print $filehandle $record_pointer -> {'address_id'.$t};
   print $filehandle $record_pointer -> {'address_type'.$t} ;
   print $filehandle $record_pointer -> {'address_status_code'.$t} ;
   print $filehandle $record_pointer -> {'address_begin_date'.$t} ;
   print $filehandle $record_pointer -> {'address_end_date'.$t} ;
```

```
    print $filehandle $record_pointer -> {'address_line1'.$t} ;
    print $filehandle $record_pointer -> {'address_line2'.$t} ;
    print $filehandle $record_pointer -> {'address_line3'.$t} ;
    print $filehandle $record_pointer -> {'address_line4'.$t} ;
    print $filehandle $record_pointer -> {'address_line5'.$t} ;
    print $filehandle $record_pointer -> {'city'.$t} ;
    print $filehandle $record_pointer -> {'state'.$t} ;
    print $filehandle $record_pointer -> {'zip_post'.$t} ;
    print $filehandle $record_pointer -> {'country'.$t} ;
    print $filehandle $record_pointer -> {'phone_primary'.$t} ;
    print $filehandle $record_pointer -> {'phone_mobile'.$t} ;
    print $filehandle $record_pointer -> {'phone_fax'.$t} ;
    print $filehandle $record_pointer -> {'phone_other'.$t} ;
    print $filehandle $record_pointer -> {'date_add_update'.$t} ;
  }
    print $filehandle "\n";

}
```

Report