```perl
#!/usr/bin/env perl
# $Id: Sretain,v 1.32 2012/08/07 23:06:08 murakami Exp murakami $
# See perl pod documentation at the end of this file

use strict;
use warnings;
use Pod::Usage;
use POSIX;
use Time::Local qw(timelocal);
use Getopt::Long;

use constant PROGRAM => "Sretain";
use constant VERSION => "1.7";

##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
##
## PROTOTYPES
##
##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

sub srb_date;
sub get_user_type;
sub convert_dwmy;
sub parse_args;
sub verify_project;
sub _error;

##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
##
## GLOBALS
##
##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

my %SRB_AUTH_INFO;
my @OBJECTS;
my @SCHEME_PARAMS;
my $CMD;
my $DAYS;
my $ARG;
my $LIST_SCHEME = 0;
my $DEFAULT_SCHEME = "Admin";
my $ERRORS = 0;
my $RECURSIVE = 0;
my $STATUS;
my $DEBUG = 0;
my $VERBOSE = 0;
my $DATE_STR;
my $CURR_TIME = time();
my %CURR_DATE =
    (
      YEAR => ((localtime($CURR_TIME))[5] + 1900),
      MONTH => ((localtime($CURR_TIME))[4] + 1),
      DAY => (localtime($CURR_TIME))[3]
    );

# List of metadata parameters
my %OPTION_PARAMS =
```

```
  (

    ADMIN_HOLD =>
      {
        SUPER => 1,
        VALUE => undef,
        SCHEME => "Admin",
        COLUMN => "Admin_Hold",
      },

    NEXT_REVIEW =>
      {
        SUPER => 0,
        VALUE => undef,
        SCHEME => "Admin",
        COLUMN => "Next_Review_Time",
      },

    PROJECT =>
      {
        SUPER => 0,
        VALUE => undef,
        SCHEME => "Admin",
        COLUMN => "HPCMP_Project_ID",
      },

    PRE_WARNING =>
      {
        SUPER => 0,
        VALUE => undef,
        SCHEME => "Admin",
        COLUMN => "PreWarning_Period",
      },

    RETENTION_DAYS =>
      {
        SUPER => 0,
        VALUE => undef,
        SCHEME => "Admin",
        COLUMN => "Retention_Period",
      },

    RETAIN_THROUGH =>
      {
        SUPER => 0,
        VALUE => undef,
        SCHEME => "Admin",
        COLUMN => "Retain_Time",
      },

    ARCHIVE =>
      {
        SUPER => 0,
        VALUE => undef,
        SCHEME => "Admin",
        COLUMN => "Archive_Behavior",
      },
```

```
115
116     IMMEDIATE =>
117       {
118         SUPER => 0,
119         VALUE => undef,
120         SCHEME => "Admin",
121         COLUMN => "Archive_Behavior",
122       },
123
124     DR =>
125       {
126         SUPER => 0,
127         VALUE => undef,
128         SCHEME => "Admin",
129         COLUMN => "DR_Behavior",
130       },
131
132     PURGE =>
133       {
134         SUPER => 0,
135         VALUE => undef,
136         SCHEME => "Admin",
137         COLUMN => "Purge_Behavior",
138       },
139   );
140
141 ##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
142 ##
143 ## BEGIN
144 ##
145 ##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
146
147 if (! parse_args())
148   {
149     _error("failed to parse arguments");
150     exit(1);
151   }
152
153 ## Exit if $DEBUG is set before actually modifying files
154 if ($DEBUG)
155   {
156     exit(0);
157   }
158
159 # Set scheme info on all requested objects
160 $CMD = 'Sscheme'; # Use native SRB Sscheme command
161 if ($RECURSIVE)
162   {
163     $CMD .= " -R";  # request recusion
164   }
165
166 my $OBJSTR = join(' ', @OBJECTS);
167 if ($OBJSTR =~ m/\*/)                          # If wildcard
168   {
169     printf("DEBUG: %s\n", $OBJSTR) if ($VERBOSE);
170     $OBJSTR =~ s/\*/\\*/g;                     # Escape wildcard
171     printf("DEBUG: %s\n", $OBJSTR) if ($VERBOSE);
```

```
172   }
173
174 $CMD .= sprintf(" -w -val '%s' %s",
175    join(',', @SCHEME_PARAMS),
176    $OBJSTR);
177
178 printf("DEBUG: %s\n", $CMD) if ($VERBOSE);
179
180 $STATUS = system($CMD);
181
182 if (($STATUS >> 8) != 0)
183   {
184     $ERRORS++;
185   }
186
187 if ($ERRORS)
188   {
189     exit(1);
190   }
191
192 ## List scheme elements if requested
193 if ($LIST_SCHEME)
194   {
195     foreach (@OBJECTS)
196       {
197         $CMD = sprintf("Sscheme %s -l -scheme \"%s\" %s",
198             $RECURSIVE ? "-r" : "",
199             $DEFAULT_SCHEME,
200             $_);
201
202         system($CMD);
203       }
204   }
205
206 exit(0);
207
208 ##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
209 ##
210 ## SUBROUTINES
211 ##
212 ##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
213
214 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
215 #
216 # NAME
217 #      _debug -- print debug information
218 #
219 # SYNOPSIS
220 #      _debug($fmt, $arg1[, $arg2]);
221 #
222 # DESCRIPTION
223 #      If $VERBOSE is set to 1, prints $fmt and optional $args using printf to
224 #      STDOUT.
225 #
226 # RETURN VALUES
227 #      None.
228 #
```

```
229 #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
230 sub _debug
231 {
232   my $fmt = shift(@_);
233   my @args = @_;
234   my $msg;
235
236   return if (! $VERBOSE);
237
238   $fmt = sprintf("DEBUG: %s%s", $fmt, ($fmt !~ /\n$/ ? "\n" : ""));
239
240   $msg = sprintf($fmt, @args);
241
242   printf($msg);
243 }
244
245 #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
246 #
247 # NAME
248 #      _error -- print error information
249 #
250 # SYNOPSIS
251 #      _error($fmt, $arg1[, $arg2]);
252 #
253 # DESCRIPTION
254 #      Prints $fmt and options $args using printf to STDERR.
255 #
256 # RETURN VALUES
257 #      None.
258 #
259 #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
260 sub _error
261 {
262   my $fmt = shift(@_);
263   my @args = @_;
264   my $msg;
265
266   $fmt = sprintf("Sretain ERROR: %s%s", $fmt, ($fmt !~ /\n$/ ? "\n" : ""));
267
268   $msg = sprintf($fmt, @args);
269
270   printf(STDERR $msg);
271 }
272
273 #:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
274 #
275 # NAME
276 #      parse_args -- parse command line arguments
277 #
278 # SYNOPSIS
279 #      $status = parse_args();
280 #
281 # DESCRIPTION
282 #      The parse_args subroutine parses command line arguments provided by the
283 #      @ARGV array.
284 #
285 # RETURN VALUES
```

```
286 #      Upon successful completion a value of 1 is returned.  Otherwise, a value
287 #      of 0 is returned.
288 #
289 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
290 sub parse_args
291 {
292   my %args;
293   my $arg;
294   my $status;
295 
296   GetOptions
297     (
298       \%args,
299       "debug",
300       "verbose",
301       "next_review=s",
302       "hold!",
303       "archive!",
304       "immediate!",
305       "list|l",
306       "purge!",
307       "dr!",
308       "project|p=s",
309       "recursive|r",
310       "version|v" => sub { printf("%s %s\n", PROGRAM, VERSION); exit(0); },
311       "help|h" => sub { pod2usage(-exitval => 0, -verbose => 1); },
312     ) or pod2usage(2);
313 
314   if ($#ARGV < 1)
315     {
316       _error("must specify retention days and one or more object or
    collection");
317       pod2usage(2);
318     }
319 
320   # Get retention period
321   $arg = shift(@ARGV);
322 
323   if ($arg =~ /^\d+[d|w|m|y]?$/)
324     {
325       $OPTION_PARAMS{RETAIN_THROUGH}{VALUE} = convert_dwmy($arg);
326     }
327   else
328     {
329       $OPTION_PARAMS{RETAIN_THROUGH}{VALUE} = srb_date($arg);
330     }
331 
332   if (! defined($OPTION_PARAMS{RETAIN_THROUGH}{VALUE}))
333     {
334       _error("invalid retention day specified '%s'",
335           $arg);
336       return(0);
337     }
338 
339   push(@SCHEME_PARAMS, sprintf("%s.%s::%s",
340       $OPTION_PARAMS{RETAIN_THROUGH}{SCHEME},
341       $OPTION_PARAMS{RETAIN_THROUGH}{COLUMN},
```

```
342       $OPTION_PARAMS{RETAIN_THROUGH}{VALUE}));
343
344   # Remaining args are objects/collections
345   if ($#ARGV == -1)
346     {
347       _error("must specifiy one or more objects or collections");
348       return(0);
349       pod2usage(2);
350     }
351
352   push(@OBJECTS, @ARGV);
353
354   # Parse remaining arguments
355   if (exists($args{verbose}))
356     {
357       $VERBOSE = 1;
358     }
359
360   if (exists($args{debug}))
361     {
362       $DEBUG = 1;
363     }
364
365   if (exists($args{recursive}))
366     {
367       $RECURSIVE = 1;
368     }
369
370   if (exists($args{list}))
371     {
372       $LIST_SCHEME = 1;
373     }
374
375   if (exists($args{archive}) || exists($args{immediate}))
376     {
377       if ((exists($args{archive}) && ! $args{archive}) &&
    exists($args{immediate}))
378         {
379           _error("cannot specify --noarchive and --immediate");
380           return(0);
381         }
382
383       push(@SCHEME_PARAMS, sprintf("%s.%s::%s",
384           $OPTION_PARAMS{ARCHIVE}{SCHEME},
385           $OPTION_PARAMS{ARCHIVE}{COLUMN},
386           (exists($args{immediate}) ? "immediate" :
387               ($args{archive} ? "yes" : "no"))));
388     }
389
390   if (exists($args{next_review}))
391     {
392       if ($args{next_review} =~
393           /\d{2,4}[-\/]\d{2}[-\/]\d{2,4}/)
394         {
395           $arg = srb_date($args{next_review});
396         }
397       elsif ($args{next_review} =~ /\d+[a-z]/)
```

```
398         {
399           $arg = convert_dwmy($args{next_review});
400         }
401       else
402         {
403           $arg = undef;
404         }
405
406       if (! defined($arg))
407         {
408           _error("invalid date specified for next review time '%s'",
409               $args{next_review});
410           return(0);
411         }
412
413       push(@SCHEME_PARAMS, sprintf("%s.%s::%s",
414           $OPTION_PARAMS{NEXT_REVIEW}{SCHEME},
415           $OPTION_PARAMS{NEXT_REVIEW}{COLUMN},
416           $arg));
417     }
418
419   if (exists($args{dr}))
420     {
421       push(@SCHEME_PARAMS, sprintf("%s.%s::%s",
422           $OPTION_PARAMS{DR}{SCHEME},
423           $OPTION_PARAMS{DR}{COLUMN},
424           ($args{dr} ? "yes" : "no")));
425     }
426
427   if (exists($args{purge}))
428     {
429       push(@SCHEME_PARAMS, sprintf("%s.%s::%s",
430           $OPTION_PARAMS{PURGE}{SCHEME},
431           $OPTION_PARAMS{PURGE}{COLUMN},
432           ($args{purge} ? "yes" : "no")));
433     }
434
435   # Must be "superuser" or "sysadmin" user type to set "Admin Hold"
436   if (exists($args{hold}))
437     {
438       $arg = get_user_type();
439
440       if (! defined($arg))
441         {
442           _error("failed to get user type");
443           return(0);
444         }
445
446       if ($arg ne "superuser" && $arg ne "sysadmin")
447         {
448           _error("cannot set \"Admin Hold\" as a non-admin user (%s)",
449               $arg);
450           return(0);
451         }
452
453       push(@SCHEME_PARAMS, sprintf("%s.%s::%s",
454           $OPTION_PARAMS{ADMIN_HOLD}{SCHEME},
```

```
455           $OPTION_PARAMS{ADMIN_HOLD}{COLUMN},
456           ($args{hold} ? "yes" : "no")));
457     }
458
459   if (exists($args{project}))
460     {
461       if (! verify_project($args{project}))
462         {
463           _error("invalid project code '%s'", $args{project});
464           return(0);
465         }
466
467       push(@SCHEME_PARAMS, sprintf("%s.%s::%s",
468           $OPTION_PARAMS{PROJECT}{SCHEME},
469           $OPTION_PARAMS{PROJECT}{COLUMN},
470           $args{project}));
471     }
472
473   _debug("Scheme parameters:");
474   _debug("--------------------");
475   foreach $arg (@SCHEME_PARAMS)
476     {
477       _debug("%s", $arg);
478     }
479   _debug("--------------------");
480
481   _debug("%s", join(',', @SCHEME_PARAMS));
482
483   return(1);
484 }
485
486 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
487 #
488 # NAME
489 #      get_user_type -- return user type
490 #
491 # SYNOPSIS
492 #      $user_type = get_user_type();
493 #
494 #      $user_type = get_user_type({ user => $user, scheme => $scheme });
495 #
496 # DESCRIPTION
497 #      Returns the SRB user type assigned to the current user or the user
498 #      specified by the user hash argument.  Optionally specify an
499 #      authentication scheme to specify.  Specifying the authentication scheme
500 #      may be necessary if a user has multiple authentication schemes defined.
501 #
502 # RETURN VALUES
503 #      Upon successful completion the SRB user type is returned.  Otherwise,
504 #      the undefined value is returned.
505 #
506 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
507 sub get_user_type
508 {
509   my ($args) = @_;
510   my $cmd;
511   my $user = undef;
```

```
512   my $scheme = undef;
513   my $type = undef;
514   my $pd;
515   my $buf;
516
517   if (exists($args->{user}))
518     {
519       $user = $args->{user};
520     }
521
522   if (exists($args->{scheme}))
523     {
524       $scheme = $args->{scheme};
525     }
526
527   if (! defined($user) || ! defined($scheme))
528     {
529       $cmd = "Senv";
530
531       if (! open($pd, "$cmd 2>&1 |"))
532         {
533           return(undef);
534         }
535
536       while ($buf = <$pd>)
537         {
538           chomp($buf);
539
540           _debug("\$buf = %s", $buf);
541
542           if (! defined($user) && $buf =~ /SRB_USER_NAME\s+'([^ ]*)'/)
543             {
544               $user = $1;
545             }
546
547           if (! defined($scheme) && $buf =~ /SRB_AUTH_SCHEME\s+'([^ ]*)'/)
548             {
549               $scheme = $1;
550             }
551         }
552
553       close($pd);
554     }
555
556   if (! defined($user) || ! defined($scheme))
557     {
558       return(undef);
559     }
560
561   $cmd = sprintf("SgetU -l -scheme %s %s",
562       $scheme, $user);
563
564   if (! open($pd, "$cmd 2>&1 |"))
565     {
566       return(undef);
567     }
568
```

```
569   while ($buf = <$pd>)
570     {
571       chomp($buf);
572 
573       _debug("\$buf = %s", $buf);
574 
575       if ($buf =~ /User Type:\s+([^ ]*)/)
576         {
577           $type = $1;
578         }
579     }
580 
581   close($pd);
582 
583   return($type);
584 }
585 
586 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
587 #
588 # NAME
589 #      srb_date -- returns a SRB formatted date string
590 #
591 # SYNOPSIS
592 #      $str = srb_date($date_spec);
593 #
594 # DESCRIPTION
595 #      The subroutine srb_date returns a SRB formatted date string calculated
596 #      from the argument specified.  The date formats supported are as follows:
597 #
598 #      YYYY-MM-DD or YYYY/MM/DD
599 #
600 #      MM-DD-YYYY or MM/DD/YYYY
601 #
602 #      The SRB date format is as follows:
603 #
604 #      YYYY-MM-DD HH:MM:SS.0000
605 #
606 # RETURN VALUES
607 #      Upon successful completion a date string formatted for SRB is returned.
608 #      Otherwise, the undefined value is returned.
609 #
610 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
611 sub srb_date
612 {
613   my $spec = shift(@_);
614   my ($year, $month, $day, $hour, $min, $sec);
615   my $curr_time;
616   my $srb_fmt = "%.4d-%.2d-%.2d %.2d:%.2d:%.2d";
617 
618   my @date_ents = split("[-\/]", $spec);
619 
620   if (length($date_ents[0]) == 4)
621     {
622       $year = $date_ents[0];
623       $month = $date_ents[1];
624       $day = $date_ents[2];
625     }
```

```
626   else
627     {
628       $year = $date_ents[2];
629       $month = $date_ents[0];
630       $day = $date_ents[1];
631     }
632
633   if (! $year =~ /^\d+$/ || ! $month =~ /^\d+$/ || ! $day =~ /^\d+$/)
634    {
635           _error("Invalid retention date specified: $spec");
636           exit(1);
637    }
638
639   $curr_time = time();
640
641   $hour = (localtime($curr_time))[2];
642   $min = (localtime($curr_time))[1];
643   $sec = (localtime($curr_time))[0];
644
645   eval
646    {
647      my $dummy = timelocal($sec, $min, $hour, $day, $month - 1, $year - 1900);
648    };
649
650   if (my $err = $@)
651    {
652           _error("Invalid retention date specified: $spec");
653           exit(1);
654    }
655
656   return(sprintf($srb_fmt, $year, $month, $day, $hour, $min, $sec));
657 }
658
659 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
660 #
661 # NAME
662 #      convert_dwmy -- converts day[suffix] into a SRB date formatted string.
663 #
664 # SYNOPSIS
665 #      $status = convert_dwmy($day_spec);
666 #
667 # DESCRIPTION
668 #      The subroutine convert_dwmy returns a SRB date formatted string converted
669 #      $day_spec.  The $day_spec is a number followed by an optional suffix:
670 #
671 #      d - days
672 #      w - week (7 days)
673 #      m - month
674 #      y - years
675 #
676 #      The SRB date format is as follows:
677 #
678 #      YYYY-MM-DD HH:MM:SS.0000
679 #
680 # RETURN VALUES
681 #      Upon successful completion the SRB date formatted string is returned.
682 #      If an invalid suffix is provided, the undefined value is returned.
```

```
683 #
684 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
685 sub convert_dwmy
686 {
687   my $spec = shift(@_);
688   my ($num, $unit);
689   my $days;
690   my $year;
691   my $month;
692   my $day;
693   my $hour;
694   my $min;
695   my $sec;
696   my $srb_fmt = "%.4d-%.2d-%.2d %.2d:%.2d:%.2d";
697
698   ($num, $unit) = $spec =~ /(\d+)([a-z]?)/;
699
700   if (length($unit) && $unit !~ /[d|w|m|y]/)
701     {
702       return(undef);
703     }
704
705   unit:
706     {
707       ($unit eq "d") && do
708         {
709          ($sec, $min, $hour, $day, $month, $year)
710          = localtime(time() + ($num * 24 * 60 *60));
711          $month += 1; $year += 1900;
712           last unit;
713         };
714
715       ($unit eq "w") && do
716         {
717          ($sec, $min, $hour, $day, $month, $year)
718          = localtime(time() + ($num * 7 * 24 * 60 *60));
719          $month += 1; $year += 1900;
720           last unit;
721         };
722
723       ($unit eq "m") && do
724         {
725          ($sec, $min, $hour, $day, $month, $year) = localtime();
726           $year += 1900 + int(($month + $num) / 12);
727           $month += 1 + $num  % 12;
728           last unit;
729         };
730
731       ($unit eq "y") && do
732         {
733          ($sec, $min, $hour, $day, $month, $year) = localtime();
734           $year += 1900 + $num;
735           $month += 1;
736           last unit;
737         };
738
739       do
```

```
740         {
741          ($sec, $min, $hour, $day, $month, $year)
742          = localtime(time() + ($num * 24 * 60 *60));
743          $month += 1; $year += 1900;
744           last unit;
745         }
746     }
747
748   return(sprintf($srb_fmt, $year, $month, $day, $hour, $min, $sec));
749 }
750 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
751 #
752 # NAME
753 #      verify_project - verifies a project ID is formatted correctly
754 #
755 # SYNOPSIS
756 #      $status = verify_project($id);
757 #
758 # DESCRIPTION
759 #      The subroutine verify_project checks to make sure that $id is a 13-digit
760 #      number.
761 #
762 # RETURN VALUES
763 #      If $id is valid, 1 is returned.  Otherwise, 0 is returned.
764 #
765 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
766 sub verify_project
767 {
768   my $project = shift(@_);
769
770   if (length($project) > 13 || $project !~ /[1-9A-Z]{1,13}/)
771     {
772       return(0);
773     }
774
775   return(1);
776 }
777
778 __END__
779
780 ##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
781 ##
782 ## POD DOCUMENTATION
783 ##
784 ## perldoc Sretain
785 ##
786 ##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
787
788 =pod
789
790 =head1 NAME
791
792 B<Sretain> - set SRB retention attributes.
793
794 =head1 SYNOPSIS
795
796 S<B<Sretain [options] days|date objects|collections>>
```

```
797
798 =head1 DESCRIPTION
799
800 B<Sretain> is the command line interface to set a I<"Retain_Time"> date on
801 Storage Resource Broker (SRB) objects. The B<Sretain> command updates the
802 I<"Admin"> Scheme which is a System Schmeme that is automatically applied during
803 ingestion of objects into SRB. The B<Sretain> command requires a B<days> or
804 B<date> argument for the Retain_Time date and B<objects> or
805 B<collections> to act upon. The I<"Last Review Time"> is set automatically by
806 the B<Sretain> command.
807
808 =head1 days|date
809
810 The B<days|date> parameter is a required parameter. The
811 B<days|date> parameter determines the new I<"Retain_Time"> date to be
812 set on Storage Resource Broker (SRB) objects. The B<days|date>
813 parameter can be specified in a variety of formats.
814
815 =over 4
816
817 =item B<integer[d|w|m|y]>
818
819 Specifies I<"Retain_Time"> date in integer days of duration from the current
820 date.  Modifiers are available for days B<d>, weeks B<w>, months B<m> and years
821 B<y>.  Days is the default if no modified is specified.
822
823 =item B<yyyy/mm/dd>
824
825 Specifies a fixed date for the I<"Retain_Time"> date in
826 year/month/day format.
827
828 =item B<yyyy-mm-dd>
829
830 Specifies a fixed date for the I<"Retain_Time"> date in
831 year-month-day format.
832
833 =item B<mm/dd/yyyy>
834
835 Specifies a fixed date for the I<"Retain_Time"> date in
836 month/day/year format.
837
838 =item B<mm-dd-yyyy>
839
840 Specifies a fixed date for the I<"Retain_Time"> date in
841 month-day-year format.
842
843 =back
844
845 =head1 objects|collections
846
847 The B<objects|collections> parameter is a required parameter. More than
848 one B<object> and/or B<collection> can be specified. The
849 B<objects|collections> parameter specifies the SRB objects and/or
850 collections to act upon.
851
852 =head1 OPTIONS
853
```

```
854 The B<Sretain> command allows a number of options to set I<"Admin"> Scheme
855 attributes besides the I<"Retain_Time"> date and includes the B<-r> option
856 which causes the attributes to be set recursively on objects within the
857 collection[s].
858
859 The options are as follows:
860
861 =over 4
862
863 =item B<-R>, B<-recursive>
864
865 Specifies that the operation be applied recursively on objects within the
866 collection[s]. The B<-R> option only works when collections are specified.
867
868 =item B<-next_review=days|date>
869
870 Specify the B<date> or number of B<days> until the next review.
871
872 =item B<-archive|-immediate|-noarchive>
873
874 Specifies whether the object is archived, archived immediately, or not archived
875 (I<"Archive Behavior"> attribute).
876
877 =item B<-dr|-nodr>
878
879 Specifies whether the object is sent to the DR system (I<"DR Behavior">
880 attribute).
881
882 =item B<-purge|-nopurge>
883
884 Specifies whether the object is released after archive. (I<"Purge Behavior">
885 attribute).
886
887 =item B<-next_review days|date>
888
889 Specifies the next review time in either B<days> or B<date> in the format
890 specified earlier.
891
892 =item B<-hold>
893
894 Specifies that the I<"Admin Hold"> column be set to yes.  B<This is an
895 administrator only option.>
896
897 =item B<-p HPCMP_Project_ID>
898
899 Specifies the 13 character HPCMP project ID that this object is associated with.
900
901 =item B<-help>
902
903 Print this help message and exit.
904
905 =item B<-version>
906
907 Print the program version and exit.
908
909 =item B<-verbose>
910
```

```
911 Print more verbose output during the execution of B<Sretain>.
912
913 =back
914
915 =head1 ADMIN SCHEME
916
917 The Admin scheme is a System Scheme that is automatically applied during
918 ingestion of objects into SRB. The Admin scheme has the following attributes:
919
920 =over 4
921
922 =item I<Archive_Behavior>
923
924 Specifies whether the object is archived, archived immediately, or not archived.
925
926 =item I<DR_Behavior>
927
928 Specifies whether the object is sent to the DR system.
929
930 =item I<Purge_Behavior>
931
932 Specifies whether the object is released after archive.
933
934 =item I<Retention_Period>
935
936 Identifies the number of days the object is retained in the archive.
937
938 =item I<Retain_Time>
939
940 Identifies the date the object will be retained through.
941
942 =item I<PreWarning_Period>
943
944 Identifies the number of days prior to the end of the retention period that the
945 user should be warned.
946
947 =item I<Last_Review_Time>
948
949 Identifies when the object was last reviewed. Automatically updated by
950 B<Sretain>.
951
952 =item I<Next_Review_Time>
953
954 Identifies when the archive policy is due to be reviewed by the user.
955
956 =item I<Admin_Hold>
957
958 Identifies whether the object is prevented from expiration and purge (not user
959 modifiable).
960
961 =item I<HPCMP_Project_ID>
962
963 Identifies the HPCMP project that the object or collection belongs to.
964 The value must be uppercase alphanumeric.
965
966 =back
967
```

```
=head1 Metadata Attribute Population Algorithm - Step Description

=over 4

=item 1.

If the attribute is a string list then the value must be one of the strings
specified for the string list.  The system uses values supplied directly by the
user (does not apply for System Master Schemes).

=item 2.

If the user does not supply a value, i.e. the value is "null", then the system
uses the user default value associated with the attribute column.

=item 3.

If there is no user default value, the system uses the administrator default
value associated with the attribute column.

=item 4.

If there is no administrator default value, the system inherits the value from
the Parent Collection.

=item 5.

If none of the above conditions are met then the system leaves the value as
"null". (From SRB User guide (2010 Alpha version)

=back

=head1 EXAMPLES

=over 4

=item Note: The following examples are based on the current date of: 2011-05-06
15:24:00

The Sretain command updates the I<"Retain_Time"> date based on the
B<days|date> required parameter.  The Sretain command calculates the date used
based on the current date if an integer[d|w|m|y] specification is made.
The current time is used for the time portion of the new I<"Retain_Time">
date time value.

=item Example 1 Setting retention time to 90 days

% S<Sretain 90 MyObj>

This command will set the retention period for the object MyObj in the current
collection to 90 days from the current date.

The equivalent native Sscheme command is:

% S<Sscheme -w -val 'Admin.Retain_Time::2011-08-04 15:24:00' MyObj>

=item Example 2 Setting retention time to 3 months
```

```
1024
1025 % S<Sretain 3m MyObj>
1026
1027 This command will set the retention period for the object MyObj in the current
1028 collection to 3 months from the current date.
1029
1030 The equivalent native Sscheme command is:
1031
1032 % S<Sscheme  -w -val 'Admin.Retain_Time::2011-08-06 15:24:00'  MyObj>
1033
1034 =item Example 3 Setting retention time to 90 days for an object in a collection
1035
1036 % S<Sretain 90d MyCol/MyObj>
1037
1038 This command will set the retention period for the object MyObj in the
1039 collection MyCol to 90 days from the current date.
1040
1041 The equivalent native Sscheme command is:
1042
1043 % S<Sscheme  -w -val 'Admin.Retain_Time::2011-08-04 15:24:00' MyCol/MyObj>
1044
1045 =item Example 4 Setting retention time to 6 months using a regular expression.
1046
1047 % S<Sretain 6m 'MyObj*'>
1048
1049 This command will set the retention period for any objects matching 'MyObj*' in
1050 the current collection to 6 months from the current date.
1051
1052 The equivalent native Sscheme command is:
1053
1054 % S<Sscheme -w -val 'Admin.Retain_Time::2011-11-06 15:24:00' 'MyObj*'>
1055
1056 Note that the regular expression must be quoted so that it will pass to SRB
1057 without being expanded by the shell.
1058
1059 =item Example 5 Setting retention time to a specfic date recursively on a
     collection
1060
1061 % S<Sretain -R 05/20/2011 MyCol>
1062
1063 This command will set the retention period for the collection MyCol and all
1064 objects in the collection recursively to 05/20/2011
1065
1066 The equivalent native Sscheme command is:
1067
1068 % S<Sscheme -w -R -val 'Admin.Retain_Time::2011-05-20 15:24:00' MyObj>
1069
1070 =item Example 6 Setting retention time and Project in a single command.
1071
1072 % S<Sretain 4w -p HPCMO92330SIS MyObj>
1073
1074 This command will set the retention period
1075 to 4 weeks from the current date and set the project to HPCMO92330SIS
1076 for the object MyObj in one command.
1077
1078 % S<Sscheme  -w -val 'Admin.Retain_Time::2011-06-03
     15:24:00,Admin.HPCMP_Project_ID::HPCMO92330SIS'  MyObj>
```

```
1079
1080 =back
1081
1082 =head1 SEE ALSO
1083
1084 B<Sreview(1)>, B<Sscheme(1)>, B<Sls(1)>, B<Senv(1)>
1085
1086 =head1 RELEASE NOTES
1087
1088 =over 4
1089
1090 =item Version 1.0 - July 23, 2010
1091
1092 Initial release.
1093
1094 =item Version 1.1 - December 17, 2010 - LAM@ARSC
1095
1096 Modified to accept uppercase alphanumberic project and short project option.
1097 Modified pod examples to match implemented syntax.
1098
1099 =item Version 1.2 - April 27, 2011 - LAM@ARSC
1100
1101 Modified for changes to scheme names and columns and changes to Scheme syntax.
1102 Removed Last_Review_Time from generated Scheme command.
1103 Remove fractional parts of integer for Retention_Period.
1104
1105 =item Version 1.3 - May 6, 2011 - LAM@ARSC
1106
1107 Removed convert_date prototype for subroutine not defined or used.
1108 Fix code to default to days (don't require the d).
1109 Removed setting of SRB_HOME and modification of PATH which was causing
1110 a 'Use of uninitialized value in concatenation' error on systems where the
1111 hard coded default path was not the path where SRB was installed.
1112 Report an error if at least two parameters are not supplied.
1113 Modified code to update Retain_Time rather than Retention_Period.
1114 Replaced the convert_days subroutine with convert_dwmy subroutine that returns
1115  a SRB date formatted string.  Removed subroutines that are no longer required.
1116 Truncate the date when used for Retain_Time to avoid SRB error.
1117 Modified date error handling - exit with error if unable to parse date.
1118
1119 =item Version 1.4 - May 10, 2011 - LAM@ARSC
1120
1121 Modified srb_date to use a different date format accepted for both
1122 Retain_Time and Next_Review_Time.  Removed truncation of Retain-Through date.
1123
1124 =item Version 1.5 - May 2, 2012 - LAM@HTL
1125
1126 Modified for change of Retain-Through attribute name to new Retain_Time name.
1127 Changed version to 1.5 alpha.
1128
1129 =item Version 1.6 - July 6, 2012 - LAM@HTL
1130
1131 Modified script to submit one 'Sscheme -w' command for all objects specified.
1132 Reported version is now 1.6 (without alpha).
1133
1134 =item Version 1.7 - August 7, 2012 - LAM@HTL
1135
```

```
1136 Modified help and man page to specify "-R" instead of "-r" for recursion to
     match other Scommands.
1137 Modified to always escape any asterisk wildcard characters in the
     objects|collections argument.
1138
1139 =back
1140
1141 =cut
1142
```