```
**********************************************************
   31091 Fri Jan  9 18:58:57 2015
new/usr/src/lib/libbsm/auditxml
5516 perl problems in libbsm/auditxml
Reviewed by: Richard Lowe <richlowe@richlowe.net>
**********************************************************
```

```
   1 #!/usr/perl5/bin/perl -w
   2 #
   3 # CDDL HEADER START
   4 #
   5 # The contents of this file are subject to the terms of the
   6 # Common Development and Distribution License (the "License").
   7 # You may not use this file except in compliance with the License.
   8 #
   9 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  10 # or http://www.opensolaris.org/os/licensing.
  11 # See the License for the specific language governing permissions
  12 # and limitations under the License.
  13 #
  14 # When distributing Covered Code, include this CDDL HEADER in each
  15 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  16 # If applicable, add the following below this CDDL HEADER, with the
  17 # fields enclosed by brackets "[]" replaced with your own identifying
  18 # information: Portions Copyright [yyyy] [name of copyright owner]
  19 #
  20 # CDDL HEADER END
  21 #
  22 #
  23 # Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
  24 # Use is subject to license terms.
  25 #

  27 # auditxml takes the audit record description (.xml file) and
  28 # generates the files needed for the C audit api.

  30 my $prog = $0; $prog =~ s|.*/||g;
  31 my $usage = <<EOF;

  33 Usage: $prog [options] <xml-input-file>
  34 Options:
  35         -d      Enable debug output
  36         -e pfx  Internal event prefix (default: AUE)
  37         -i pfx  Interface prefix (default: adt)
  38                 External event prefix is uppercase version of this string.
  39         -o dir  Output directory (default: current dir)

  41 EOF

  43 use auditxml;
  44 use Getopt::Std;
  45 use strict;

  47 our $debug = 0; # normal use is to set via the file being parsed.
  48                # <debug set="on"/> or <debug set="off"/> or <debug/>
  49                # if the set attribute is omitted, debug state is toggled
  50                # Override with appDebug, but toggle won't do what you
  51                # want.
  52 my $appDebug = 0; # used after return from "new auditxml";

  54 # Process command-line options
  55 our ($opt_d, $opt_e, $opt_i, $opt_o);
  56 $opt_e = "";
  57 $opt_i = "";
  58 $opt_o = "";
  59 if (!getopts('de:i:o:') || $#ARGV != 0) {
  60     die $usage;
```



```
  61 }
______unchanged_portion_omitted_

 575 sub generateTableC {
 576     my $event = shift;
 577     my $eventId = shift;
 578     my $eventType = shift;
 579     my $eventHeader = shift;
 580     my $omit = shift;

 582     my %tokenType = (
 583         #
 584         #       tokenTypes are the ones that are actually defined
 585         #       for use in adt.xml audit records
 586         #

 588         #         'acl'                 => 'AUT_ACL',           # not defined
 589         #         'arbitrary'           => 'AUT_ARBITRARY',     # not defined
 590         #         'arg'                 => 'AUT_ARG',           # not defined
 591         #         'attr'                => 'AUT_ATTR',
 592                   'command'             => 'AUT_CMD',
 593                   'command_alt'         => 'ADT_CMD_ALT',       # dummy token id
 594         #         'date'                => 'AUT_TEXT',          # not used
 595         #         'exec_args'           => 'AUT_EXEC_ARGS',     # not defined
 596         #         'exec_env'            => 'AUT_EXEC_ENV',      # not defined
 597         #         'exit'                => 'AUT_EXIT',          # not defined
 598                   'fmri'                => 'AUT_FMRI',
 599         #         'groups'              => 'AUT_GROUPS',        # not defined
 600         #         'header'              => 'AUT_HEADER',        # not defined
 601                   'in_peer'             => 'ADT_IN_PEER',       # dummy token id
 602                   'in_remote'           => 'ADT_IN_REMOTE',     # dummy token id
 603         #         'ipc'                 => 'AUT_IPC',           # not defined
 604         #         'ipc_perm'            => 'AUT_IPC_PERM',      # not defined
 605                   'iport'               => 'AUT_IPORT',
 606                   'label'               => 'AUT_LABEL',
 607                   'newgroups'           => 'AUT_NEWGROUPS',
 608         #         'opaque'              => 'AUT_OPAQUE',        # not defined
 609                   'path'                => 'AUT_PATH',
 610                   'path_list'           => '-AUT_PATH',         # dummy token id
 611                   'process'             => 'AUT_PROCESS',
 612                   'priv_effective'      => 'ADT_AUT_PRIV_E',    # dummy token id
 613                   'priv_limit'          => 'ADT_AUT_PRIV_L',    # dummy token id
 614                   'priv_inherit'        => 'ADT_AUT_PRIV_I',    # dummy token id
 615                   'return'              => 'AUT_RETURN',
 616         #         'seq'                 => 'AUT_SEQ',           # not defined
 617         #         'socket'              => 'AUT_SOCKET',        # not defined
 618         #         'socket-inet'         => 'AUT_SOCKET_INET',
 619                   'subject'             => 'AUT_SUBJECT',
 620                   'text'                => 'AUT_TEXT',
 621                   'tid'                 => 'AUT_TID',
 622         #         'trailer'             => 'AUT_TRAILER',       # not defined
 623                   'uauth'               => 'AUT_UAUTH',
 624                   'user'                => 'AUT_USER',
 625                   'zonename'            => 'AUT_ZONENAME'
 626                  );

 628     my @xlateEntryList = ();

 630     my $external = $event->getExternal();
 631     my $internal = $event->getInternal();

 633     unless ($external) {
 634         print STDERR "No external object captured for event $eventId\n";
 635         return;
 636     }
 637     if ($eventType) {
```

```
638         $nameTranslation{$eventId} = $eventId;
639     } else {
640         $nameTranslation{$eventId} = $external->getInternalName();
641     }
642     unless ($internal) {
643         print STDERR "No internal object captured for event $eventId\n";
644         return;
645     }
646     my @entryRef = $internal->getEntries();
647     my $entryRef;
648     my @tokenOrder = ();
649     my $firstTokenIndex = 0; # djdj not used yet, djdj BUG!
650                              # needs to be used by translate table

652     if ($internal->isReorder()) { # prescan the entry list to get the token orde
653       my @inputOrder;
654       foreach $entryRef (@entryRef) {
655         my ($intEntry, $entry) = @$entryRef;
656         push (@inputOrder, $intEntry->getAttr('order'));
657       }

659       my $i; # walk down the inputOrder list once
660       my $k = 1; # discover next in line
661       my $l = 0; # who should point to next in line
662       for ($i = 0; $i <= $#inputOrder; $i++) {
663         my $j;
664         for ($j = 0; $j <= $#inputOrder; $j++) {
665           if ($k == $inputOrder[$j]) {
666             if ($k == 1) {
667                 $firstTokenIndex = $j;
668             } else {
669                 $tokenOrder[$l] = "&(selfReference[$j])";
670             }
671             $l = $j;
672             last;
673           }
674         }
675         $k++;
676       }
677       $tokenOrder[$l] = 'NULL';
678     }
679     else { # default order -- input order same as output
680       my $i;
681       my $j;
682       for ($i = 0; $i < $#entryRef; $i++) {
683         my $j = $i + 1;
684         $tokenOrder[$i] = "&(selfReference[$j])";
685       }
686       $tokenOrder[$#entryRef] = 'NULL';
687     }

689     my $sequence = 0;
690     foreach $entryRef (@entryRef) {
691       my ($intEntry, $entry) = @$entryRef;
692       my $entryId = $entry->getAttr('id');

694       my ($extEntry, $unusedEntry, $tokenId) =
695         $external->getEntry($entryId);
696       my $opt = $extEntry->getAttr('opt');

698       if ($opt eq 'none') {
699         if (defined ($doc->getToken($tokenId))) {
700           if (defined ($tokenType{$tokenId})) {
701             $tokenId = $tokenType{$tokenId};
702           }
703           else {
```



```
704             print STDERR "token id $tokenId not implemented\n";
705           }
706         }
707         else {
708           print STDERR "token = $tokenId is undefined\n";
709           $tokenId = 'error';
710         }
711         my ($xlate, $jni) =
712           formatTableEntry ('', $tokenId, $eventId, '', 0, 0,
713                             $tokenOrder[$sequence], 'NULL', '', $omit);
710                             $tokenOrder[$sequence], 'NULL', $omit);
714         push (@xlateEntryList, $xlate);
715       }
716       else {
717         my $dataType = $extEntry->getAttr('type');
718         $dataType =~ s/\s+//g;   # remove blanks (char * => char*)

720         my $enumGroup = '';
721         if ($dataType =~ /^msg/i) {
722             $enumGroup = $dataType;
723             $enumGroup =~ s/^msg\s*//i;
724             $enumGroup = "${pfx_adt}_" . $enumGroup;
725         }
726         my $required = ($opt eq 'required') ? 1 : 0;
727         my $tsol = 0;
728         my $tokenId = $intEntry->getAttr('token');
729         my $token;
730         my $tokenName;
731         my $tokenFormat = $intEntry->getAttr('format');
732         if (defined ($tokenFormat)) {
733           $tokenFormat = "\"$tokenFormat\"";
734         }
735         else {
736           $tokenFormat = 'NULL';
737         }
738
739         if (defined ($token = $doc->getToken($tokenId))) {
740           $tsol = (lc $token->getUsage() eq 'tsol') ? 1 : 0;
741           if (defined ($tokenType{$tokenId})) {
742             $tokenName = $tokenType{$tokenId};
743           }
744           else {
745             print STDERR "token id $tokenId not implemented\n";
746           }
747         }
748         else {
749           print STDERR
750             "$tokenId is an unimplemented token ($entryId in $eventId)\n";
751           $tokenName = 'AUT_TEXT';
752         }
753         my ($xlate, $jni) =
754           formatTableEntry($entryId, $tokenName, $eventId, $dataType, $required,
755                            $tsol, $tokenOrder[$sequence], $tokenFormat,
756                            $enumGroup, $omit);
757         push (@xlateEntryList, $xlate);
758       }
759       $sequence++;
760     }
761     $xlateEventTable{$eventId} = [\@xlateEntryList, $eventType, $firstTokenIndex
762                                   $eventHeader];
763 }
```

*_____unchanged_portion_omitted_*