# build-xlsx-1.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245 | #!/usr/bin/env python3  #  # build-xlsx - Generate a spread sheet from files  #  # USAGE  #  # (1) Output an empty sheet  #  #　 $ build-xlsx -o config.xlsx  #  # (2) Generate a filled sheet  #  # $ build-xlsx esr60.txt esr68.txt verify-targets-to-chapters.csv  #  import re  import sys  import glob  import getopt  import csv  import os  BASEDIR = os.path.dirname(os.path.realpath(\_\_file\_\_))  sys.path.append(BASEDIR)  import adlib  try:  import xlsxwriter  except ImportError:  print('ERROR: Please install xlswriter to run this script\n')  print(' $ sudo apt install python3-xlsxwriter\n')  sys.exit(1)  #  # Global settings  ESR\_PREVIOUS = 'esr60'  ESR\_CURRENT = 'esr68'  CHAPTERS\_CSV = 'verify-targets-to-chapters.csv'  WORKBOOK\_DEF = [  ('基本設定', [  'Install',  'Application',  'Admin',  'Security',  'Privacy',  'Startup',  'Websearch',  'Location',  'Download',  'Tab',  'Network',  'Update',  'Ui',  'Script',  'Plugin',  'External',  'Stability',  'Appearance',  'Performance',  'Addon-IEView',  'Addon-FireIE',  'Addon-Acrobat',  ]),  ('機能無効化', [  'MenuShortcut',  ]),  ]  DEFAULT\_FORMAT = {  'valign': 'top',  'border': 1,  'font\_size': 8,  'font\_name': 'MS Gothic',  'text\_wrap': 1  }  #  # XLSX writer  def is\_deprecated(x):  return '廃止' in x  def count\_options(conf):  return sum(len(item['opts']) for item in conf)  def create\_formats(wb):  def new\_format(\*\*kwargs):  return wb.add\_format(dict(DEFAULT\_FORMAT, \*\*kwargs))  return {  'default': new\_format(),  'noborder': new\_format(border=0),  'center': new\_format(align='center'),  'deprecated': new\_format(bg\_color='#dddddd'),  'question': new\_format(bg\_color='#90ee90'),  'selected': new\_format(bg\_color='#fffa95'),  'selected\_changed': new\_format(bg\_color='#ffb571'),  }  def write\_legend(sheet, formats, row):  sheet.write(row, 1, '', formats['selected'])  sheet.write(row, 2, '前バージョンから引き続き利用する項目', formats['noborder'])  sheet.write(row + 1, 1, '', formats['selected\_changed'])  sheet.write(row + 1, 2, '前バージョンから異同がある項目', formats['noborder'])  sheet.write(row + 2, 1, '', formats['deprecated'])  sheet.write(row + 2, 2, '廃止済みの項目', formats['noborder'])  def write\_header(sheet, formats):  curr = ESR\_CURRENT.upper()  prev = ESR\_PREVIOUS.upper()  fmt = formats['center']  sheet.freeze\_panes(1, 0)  sheet.write(0, 0, 'カテゴリー', fmt)  sheet.write(0, 1, '項目設定番号', fmt)  sheet.write(0, 2, 'カスタマイズ項目 (目的)', fmt)  sheet.write(0, 3, '状態', fmt)  sheet.write(0, 4, '選択肢番号', fmt)  sheet.write(0, 5, '選択肢', fmt)  sheet.write(0, 6, '設定内容の雛形\n(%s)' % curr, fmt)  sheet.write(0, 7, '最終的に反映した設定値\n(%s)' % curr, fmt)  sheet.write(0, 8, '%s→%sでの変更' % (prev, curr), fmt)  sheet.write(0, 9, '検証手順書対応番号', fmt)  sheet.write(0, 11, '設定内容の雛形\n(%s)' % prev, fmt)  sheet.write(0, 12, '最終的に反映した設定値\n(%s)' % prev, fmt)  sheet.set\_row(0, 25)  sheet.set\_column(0, 12, None, formats['default'])  sheet.set\_column(0, 0, 10)  sheet.set\_column(1, 1, 10)  sheet.set\_column(2, 2, 30)  sheet.set\_column(3, 3, 5)  sheet.set\_column(4, 4, 5)  sheet.set\_column(5, 5, 20)  sheet.set\_column(6, 6, 40)  sheet.set\_column(7, 7, 40)  sheet.set\_column(8, 8, 10)  sheet.set\_column(9, 9, 10)  sheet.set\_column(10, 10, 12)  sheet.set\_column(11, 11, 40)  sheet.set\_column(12, 12, 40)  def generate\_xlsx(wb, conf\_curr, conf\_prev, chapters, excludes):  formats = create\_formats(wb)  for title, files in WORKBOOK\_DEF:  if title in excludes:  continue  sheet = wb.add\_worksheet(title)  write\_header(sheet, formats)  row = 1  for fn in files:  curr = adlib.load(os.path.join(BASEDIR, ESR\_CURRENT, fn))  prev = adlib.load\_as\_dict(os.path.join(BASEDIR, ESR\_PREVIOUS, fn))  sheet.merge\_range(row, 0, row + count\_options(curr) - 1, 0, '')  for item in curr:  if len(item['opts']) > 1:  sheet.merge\_range(row, 1, row + len(item['opts']) - 1, 1, '')  sheet.merge\_range(row, 2, row + len(item['opts']) - 1, 2, '')  for opt in item['opts']:  selected = ''  status = ''  chapter = ''  fmt = formats['default']  item\_fmt = formats['default']  opt\_id = opt['opt\_id']  if is\_deprecated(item['item\_title']):  item\_fmt = formats['deprecated']  fmt = formats['deprecated']  elif is\_deprecated(opt['opt\_title']):  fmt = formats['deprecated']  elif opt\_id in conf\_curr:  selected = 'y'  chapter = chapters.get(opt\_id, '省略')  if opt\_id not in conf\_prev:  fmt, status = formats['selected\_changed'], '新規'  elif conf\_prev[opt\_id] != conf\_curr[opt\_id]:  fmt, status = formats['selected\_changed'], '変更あり'  else:  fmt, status = formats['selected'], ''  sheet.write(row, 0, fn, formats['default'])  sheet.write(row, 1, int(item['item\_no']), item\_fmt)  sheet.write(row, 2, item['item\_title'], item\_fmt)  sheet.write(row, 3, selected, fmt)  sheet.write(row, 4, int(opt['opt\_no']), fmt)  sheet.write(row, 5, opt['opt\_title'], fmt)  sheet.write(row, 6, opt['conf'].strip(), fmt)  sheet.write(row, 7, conf\_curr.get(opt\_id, ''), fmt)  sheet.write(row, 8, status, fmt)  sheet.write(row, 9, chapter, formats['default'])  sheet.write(row, 10, '', formats['noborder'])  sheet.write(row, 11, prev.get(opt\_id, ''), fmt)  sheet.write(row, 12, conf\_prev.get(opt\_id, ''), fmt)  row += 1  write\_legend(sheet, formats, row+1)  #  # main  def load\_chapters(path):  try:  with open(path) as fp:  return dict(csv.reader(fp))  except FileNotFoundError:  return {}  def main(args):  conf\_curr = {}  conf\_prev = {}  chapters = {}  outfile = 'config.xlsx'  excludes = []  opts, args = getopt.getopt(args, 'o:x:')  for k, v in opts:  if k == '-o':  outfile = v  elif k == '-x':  excludes = v.split(',')  for arg in args:  if ESR\_CURRENT in arg:  print('%s -> %s' % (ESR\_CURRENT, arg))  conf\_curr = adlib.load\_as\_dict(arg)  elif ESR\_PREVIOUS in arg:  print('%s -> %s' % (ESR\_PREVIOUS, arg))  conf\_prev = adlib.load\_as\_dict(arg)  elif CHAPTERS\_CSV in arg:  print('Loading', os.path.basename(arg))  chapters = load\_chapters(arg)  with xlsxwriter.Workbook(outfile) as wb:  generate\_xlsx(wb, conf\_curr, conf\_prev, chapters, excludes)  print('Generated:', wb.filename)  if \_\_name\_\_ == '\_\_main\_\_':  sys.exit(main(sys.argv[1:])) |

# build-xlsx-2.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302 | #!/usr/bin/env python3  # This Source Code Form is subject to the terms of the Mozilla Public  # License, v. 2.0. If a copy of the MPL was not distributed with this  # file, You can obtain one at http://mozilla.org/MPL/2.0/.  #  # build-xlsx - Generate a spread sheet from files  #  # USAGE  #  # (1) Output an empty sheet  #  #　 $ build-xlsx -o config.xlsx  #  # (2) Generate a filled sheet  #  # $ build-xlsx esr68.txt esr78.txt ... verify-targets-to-chapters.csv  # $ build-xlsx -p esr68.txt -c esr78.txt  # $ build-xlsx -d ESR68:esr68.txt -d ESR78:esr78.txt -d "ESR78 variation:esr78-variation.txt"  #  import re  import sys  import glob  import getopt  import csv  import os  BASEDIR = os.path.dirname(os.path.realpath(\_\_file\_\_))  sys.path.append(BASEDIR)  import adlib  try:  import xlsxwriter  except ImportError:  print('ERROR: Please install xlswriter to run this script\n')  print(' $ sudo apt install python3-xlsxwriter\n')  sys.exit(1)  #  # Global settings  ESR\_PREVIOUS = 'esr78'  ESR\_CURRENT = 'esr91'  CHAPTERS\_CSV = 'verify-targets-to-chapters.csv'  WORKBOOK\_DEF = [  ('基本設定', [  'Install',  'Application',  'Admin',  'Security',  'Privacy',  'Startup',  'Websearch',  'Location',  'Download',  'Tab',  'Network',  'Update',  'Ui',  'Script',  'Plugin',  'External',  'Stability',  'Appearance',  'Performance',  'Addon-IEView',  'Addon-FireIE',  'Addon-Acrobat',  ]),  ('機能無効化', [  'MenuShortcut',  ]),  ]  DEFAULT\_FORMAT = {  'valign': 'top',  'border': 1,  'font\_size': 8,  'font\_name': 'MS Gothic',  'text\_wrap': 1  }  #  # XLSX writer  def is\_deprecated(x):  return '廃止' in x  def count\_options(conf):  return sum(len(item['opts']) for item in conf)  def sanitize\_conf(conf):  return re.sub(' \*[^:]+:\n', '', conf).strip()  def create\_formats(wb):  def new\_format(\*\*kwargs):  return wb.add\_format(dict(DEFAULT\_FORMAT, \*\*kwargs))  return {  'default': new\_format(),  'noborder': new\_format(border=0),  'center': new\_format(align='center'),  'changed': new\_format(bold=True),  'deprecated': new\_format(bg\_color='#dddddd'),  'question': new\_format(bg\_color='#90ee90'),  'selected': new\_format(bg\_color='#fffa95'),  'selected\_changed': new\_format(bg\_color='#ffb571'),  }  def write\_legend(sheet, formats, row):  sheet.write(row, 1, '', formats['selected'])  sheet.write(row, 2, '前バージョンから引き続き利用する項目', formats['noborder'])  sheet.write(row + 1, 1, '', formats['selected\_changed'])  sheet.write(row + 1, 2, '前バージョンから異同がある項目', formats['noborder'])  sheet.write(row + 2, 1, '', formats['deprecated'])  sheet.write(row + 2, 2, '廃止済みの項目', formats['noborder'])  def write\_header(sheet, formats, conf):  fmt = formats['center']  sheet.freeze\_panes(1, 0)  sheet.write(0, 0, 'カテゴリー', fmt)  sheet.write(0, 1, '項目設定番号', fmt)  sheet.write(0, 2, 'カスタマイズ項目 (目的)', fmt)  sheet.write(0, 3, '選択肢番号', fmt)  sheet.write(0, 4, '選択肢', fmt)  sheet.write(0, 5, '設定内容の雛形\n(%s)' % ESR\_CURRENT.upper(), fmt)  col\_count = 5  prev\_key = ESR\_PREVIOUS.upper()  for key in conf.keys():  if key == ESR\_PREVIOUS.upper():  continue  sheet.write(0, col\_count+1, '反映した設定値\n(%s)' % key, fmt)  sheet.write(0, col\_count+2, '%s→%sでの変更' % (prev\_key, key), fmt)  sheet.set\_column(col\_count+1, col\_count+1, 40)  sheet.set\_column(col\_count+2, col\_count+2, 10)  col\_count+=2  prev\_key = key  sheet.write(0, col\_count+1, '検証手順書対応番号', fmt)  sheet.write(0, col\_count+3, '設定内容の雛形\n(%s)' % ESR\_PREVIOUS.upper(), fmt)  sheet.write(0, col\_count+4, '反映した設定値\n(%s)' % ESR\_PREVIOUS.upper(), fmt)  sheet.set\_row(0, 25)  sheet.set\_column(0, 0, 10)  sheet.set\_column(1, 1, 10)  sheet.set\_column(2, 2, 30)  sheet.set\_column(3, 3, 5)  sheet.set\_column(4, 4, 20)  sheet.set\_column(5, 5, 40)  sheet.set\_column(col\_count+1, col\_count+1, 10)  sheet.set\_column(col\_count+2, col\_count+2, 12)  sheet.set\_column(col\_count+3, col\_count+3, 40)  sheet.set\_column(col\_count+4, col\_count+4, 40)  col\_count+=4  sheet.set\_column(0, col\_count, None, formats['default'])  def generate\_xlsx(wb, conf, chapters, excludes):  formats = create\_formats(wb)  prev\_conf = conf[ESR\_PREVIOUS.upper()]  for title, files in WORKBOOK\_DEF:  if title in excludes:  continue  sheet = wb.add\_worksheet(title)  write\_header(sheet, formats, conf)  row = 1  for fn in files:  curr = adlib.load(os.path.join(BASEDIR, ESR\_CURRENT, fn))  prev = adlib.load\_as\_dict(os.path.join(BASEDIR, ESR\_PREVIOUS, fn))  sheet.merge\_range(row, 0, row + count\_options(curr) - 1, 0, '')  for item in curr:  if len(item['opts']) > 1:  sheet.merge\_range(row, 1, row + len(item['opts']) - 1, 1, '')  sheet.merge\_range(row, 2, row + len(item['opts']) - 1, 2, '')  for opt in item['opts']:  status = ''  chapter = ''  fmt = formats['default']  item\_fmt = formats['default']  opt\_id = opt['opt\_id']  applied\_prev\_conf = prev\_conf.get(opt\_id, {'conf':''})['conf']  template\_curr\_conf = opt['conf'].strip()  template\_prev\_conf = prev.get(opt\_id, {'conf':''})['conf']  if is\_deprecated(item['item\_title']):  item\_fmt = formats['deprecated']  col\_count = 5  base\_conf = prev\_conf  applied\_base\_conf = applied\_prev\_conf  for key, variation\_conf in conf.items():  if key == key == ESR\_PREVIOUS.upper():  continue  variation\_status = ''  variation\_fmt = ''  applied\_variation\_conf = variation\_conf.get(opt\_id, {'conf':''})['conf']  if is\_deprecated(item['item\_title']) or is\_deprecated(opt['opt\_title']):  variation\_fmt = formats['deprecated']  elif opt\_id in variation\_conf:  chapter = chapters.get(opt\_id, '省略')  if opt\_id not in base\_conf:  variation\_fmt, variation\_status = formats['selected\_changed'], '新規'  elif sanitize\_conf(applied\_base\_conf) != sanitize\_conf(applied\_variation\_conf):  variation\_fmt, variation\_status = formats['selected\_changed'], '変更あり'  else:  variation\_fmt, variation\_status = formats['selected'], ''  elif base\_conf == prev\_conf:  if sanitize\_conf(template\_curr\_conf) != sanitize\_conf(template\_prev\_conf):  chapter = chapters.get(opt\_id, '省略')  if template\_prev\_conf == '':  variation\_fmt, variation\_status = formats['changed'], '新規（未設定）'  else:  variation\_fmt, variation\_status = formats['changed'], '変更あり（未設定）'  else:  if sanitize\_conf(applied\_base\_conf) != sanitize\_conf(applied\_variation\_conf):  variation\_status = '削除'  if base\_conf == prev\_conf:  fmt = variation\_fmt  sheet.write(row, col\_count+1, applied\_variation\_conf, variation\_fmt)  sheet.write(row, col\_count+2, variation\_status, variation\_fmt)  col\_count+=2  base\_conf = variation\_conf  applied\_base\_conf = applied\_variation\_conf  sheet.write(row, 0, fn, formats['default']) # A  sheet.write(row, 1, int(item['item\_no']), item\_fmt) # B  sheet.write(row, 2, item['item\_title'], item\_fmt) # C  sheet.write(row, 3, int(opt['opt\_no']), fmt) # D  sheet.write(row, 4, opt['opt\_title'], fmt) # E  sheet.write(row, 5, template\_curr\_conf, fmt) # F  sheet.write(row, col\_count+1, chapter, formats['default'])  sheet.write(row, col\_count+2, '', formats['noborder'])  sheet.write(row, col\_count+3, template\_prev\_conf, fmt)  sheet.write(row, col\_count+4, applied\_prev\_conf, fmt)  row += 1  write\_legend(sheet, formats, row+1)  #  # main  def load\_chapters(path):  try:  with open(path) as fp:  return dict(csv.reader(fp))  except FileNotFoundError:  return {}  def main(args):  conf = {}  chapters = {}  outfile = 'config.xlsx'  excludes = []  opts, args = getopt.getopt(args, 'o:x:p:c:d:')  for k, v in opts:  if k == '-o':  outfile = v  elif k == '-x':  excludes = v.split(',')  elif k == '-p':  conf[ESR\_PREVIOUS.upper()] = v  elif k == '-c':  conf[ESR\_CURRENT.upper()] = v  elif k == '-d':  parts = v.split(':', 1)  conf[parts[0]] = parts[1]  for arg in args:  if ESR\_PREVIOUS in arg and not ESR\_PREVIOUS.upper() in conf:  print('%s -> %s' % (ESR\_PREVIOUS, arg))  conf[ESR\_PREVIOUS.upper()] = arg  elif ESR\_CURRENT in arg and not ESR\_CURRENT.upper() in conf:  print('%s -> %s' % (ESR\_CURRENT, arg))  conf[ESR\_CURRENT.upper()] = arg  elif CHAPTERS\_CSV in arg:  print('Loading', os.path.basename(arg))  chapters = load\_chapters(arg)  for label, path in conf.items():  conf[label] = adlib.load\_as\_dict(path)  with xlsxwriter.Workbook(outfile) as wb:  generate\_xlsx(wb, conf, chapters, excludes)  print('Generated:', wb.filename)  if \_\_name\_\_ == '\_\_main\_\_':  sys.exit(main(sys.argv[1:])) |

# build-xlsx-3.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437  438  439  440  441  442  443  444  445  446  447  448  449  450  451  452  453  454  455  456  457  458  459  460  461 | #!/usr/bin/env python3  # This Source Code Form is subject to the terms of the Mozilla Public  # License, v. 2.0. If a copy of the MPL was not distributed with this  # file, You can obtain one at http://mozilla.org/MPL/2.0/.  #  # build-xlsx - Generate a spread sheet from files  #  # USAGE  #  # (1) Output an empty sheet  #  # $ build-xlsx -o config.xlsx  #  # (2) Generate a filled sheet  #  # $ build-xlsx esr78.txt esr91.txt ... verify-targets-to-chapters.csv  # $ build-xlsx -p esr78.txt -c esr91.txt  # $ build-xlsx -d ESR78:esr78.txt -d ESR91:esr91.txt -d "ESR91 variation:esr91-variation.txt"  #  # DEFINITION OF TERMS IN THIS MODULE  #  # For example, about "Security-9-3 about:configの利用の可否：禁止する" on Firefox ESR91:  #  # \* category: "Security", this is same to the name of the file under "esr91/"  # \* item: "Security-9"  # \* items: "Security-1", "Security-2", "Security-3", and others defined in the file "esr91/Security"  # \* option: "Security-9-1", "Security-9-2", "Security-9-3", and others  # \* config: `"BlockAboutConfig": true,` or others, defined in the given "conf" file like "esr91.txt"  # \* template: `"BlockAboutConfig": true,` or others, defined in the file "esr91/Security"  #  # \* conf: A file listing chosen options. Please note this is not an abbr of "config".  # \* curr/prev: curr=ESR91, prev=ESR78 (versions)  import re  import sys  import glob  import getopt  import csv  import os  BASEDIR = os.path.dirname(os.path.realpath(\_\_file\_\_))  sys.path.append(BASEDIR)  import adlib  try:  import xlsxwriter  except ImportError:  print('ERROR: Please install xlswriter to run this script\n')  print(' $ sudo apt install python3-xlsxwriter\n')  sys.exit(1)  #  # Global settings  ESR\_PREVIOUS = 'esr78'  ESR\_CURRENT = 'esr91'  CHAPTERS\_CSV = 'verify-targets-to-chapters.csv'  WORKBOOKS = [  ('基本設定', [  'Install',  'Application',  'Admin',  'Security',  'Privacy',  'Startup',  'Websearch',  'Location',  'Download',  'Tab',  'Network',  'Update',  'Ui',  'Script',  'Plugin',  'External',  'Stability',  'Appearance',  'Performance',  'Addon-IEView',  'Addon-FireIE',  'Addon-Acrobat',  'Addon-Skysea',  ]),  ('機能無効化', [  'MenuShortcut',  ]),  ]  DEFAULT\_FORMAT = {  'valign': 'top',  'border': 1,  'font\_size': 8,  'font\_name': 'MS Gothic',  'text\_wrap': 1,  }  CATEGORY\_COLUMNS = [ # label, width, key, format  ('カテゴリー', 10, 'category', 'default'),  ]  HEADING\_COLUMNS = [ # label, width, key, format  ('項目設定番号', 10, 'index', None),  ('カスタマイズ項目 (目的)', 30, 'title', None),  ]  LEADING\_COLUMNS = [ # label, width, key, format  ('選択肢番号', 5, 'option\_index', None),  ('選択肢', 20, 'option\_title', None),  ('設定内容の雛形\n(%s)' % ESR\_CURRENT.upper(),  40, 'template\_config', None),  ]  def variation\_columns(version, prev\_version):  return [ # label, width, key, format  ('反映した設定値\n(%s)' % version, 40, None, None),  ('%s→%sでの変更' % (prev\_version, version), 10, None, None),  ]  VERIFICATION\_COLUMNS = [ # label, width, key, format  ('検証手順書対応番号', 10, 'verification\_chapter', 'default'),  ('', 12, None, 'noborder'),  ]  PREV\_VERSION\_COLUNBS = [ # label, width, key, format  ('設定内容の雛形\n(%s)' % ESR\_PREVIOUS.upper(), 40, 'template\_prev\_config', None),  ('反映した設定値\n(%s)' % ESR\_PREVIOUS.upper(), 40, 'applied\_prev\_config', None),  ]  #  # XLSX writer  class ConfigurationSheet:  def \_\_init\_\_(self, confs, formats, sheet):  self.\_confs = confs  self.\_formats = formats  self.\_sheet = sheet  def iterate\_all\_confs(self):  return self.\_confs.items()  def write\_cell(self, row, column, contents, format):  self.\_sheet.write(row, column, contents, self.\_formats[format])  def \_set\_cell\_visual(self, row, column, width, format = None):  if format:  self.\_sheet.set\_column(row, column, width, self.\_formats[format])  else:  self.\_sheet.set\_column(row, column, width)  def write\_header(self):  sheet = self.\_sheet  sheet.freeze\_panes(1, 0)  sheet.set\_row(0, 25)  column\_offset = 0  column\_offset += self.\_write\_header\_columns(CATEGORY\_COLUMNS, 0)  column\_offset += self.\_write\_header\_columns(HEADING\_COLUMNS, column\_offset)  column\_offset += self.\_write\_header\_columns(LEADING\_COLUMNS, column\_offset)  last\_variation = ESR\_PREVIOUS.upper()  for variation in self.\_confs.keys():  if variation == ESR\_PREVIOUS.upper():  continue  columns = variation\_columns(variation, last\_variation)  column\_offset += self.\_write\_header\_columns(columns, column\_offset)  last\_variation = variation  column\_offset += self.\_write\_header\_columns(VERIFICATION\_COLUMNS, column\_offset)  column\_offset += self.\_write\_header\_columns(PREV\_VERSION\_COLUNBS, column\_offset)  def \_write\_header\_columns(self, columns, column\_offset):  for index, column in enumerate(columns):  label, width, \_key, \_format = column  self.write\_cell(0, column\_offset + index, label, 'center')  self.\_set\_cell\_visual(column\_offset + index, column\_offset + index, width)  return len(columns)  def merge\_category\_heading(self, row, items):  for index, \_column in enumerate(CATEGORY\_COLUMNS):  self.\_sheet.merge\_range(row, index, row + self.\_count\_options(items) - 1, index, '')  def \_count\_options(self, items):  return sum(len(item['options']) for item in items)  def try\_merge\_item\_heading(self, row, item):  if len(item['options']) <= 1:  return  sheet = self.\_sheet  column\_offset = len(CATEGORY\_COLUMNS)  for index, \_column in enumerate(HEADING\_COLUMNS):  sheet.merge\_range(row, column\_offset + index, row + len(item['options']) - 1, column\_offset + index, '')  def write\_legend(self, row):  self.write\_cell(row, 1, '', 'selected')  self.write\_cell(row, 2, '前バージョンから引き続き利用する項目', 'noborder')  self.write\_cell(row + 1, 1, '', 'selected\_changed')  self.write\_cell(row + 1, 2, '前バージョンから異同がある項目', 'noborder')  self.write\_cell(row + 2, 1, '', 'deprecated')  self.write\_cell(row + 2, 2, '廃止済みの項目', 'noborder')  class ConfigurationRow:  def \_\_init\_\_(self, sheet, index, item, option, category,  prev\_conf, prev\_items, verification\_chapters):  self.\_sheet = sheet  self.\_index = index  self.\_item = item  self.\_option = option  self.\_category = category  self.\_prev\_conf = prev\_conf  self.\_verification\_chapters = verification\_chapters  self.\_verification\_chapter = ''  self.\_prev\_config = self.\_get\_option\_config(self.\_prev\_conf)  self.\_template\_prev\_config = self.\_get\_option\_config(prev\_items)  self.\_template\_curr\_config = option['config'].strip()  def \_get\_option\_config(self, conf\_or\_items):  found\_option = conf\_or\_items.get(self.\_option['option\_id'])  if not found\_option:  return ''  return found\_option['config']  def write(self):  column\_offset = 0  column\_offset += self.\_write\_item\_columns(CATEGORY\_COLUMNS)  # Heading column must be written for all rows, otherwise merged cells will have  # a partial border line just for the first row.  heading\_format = 'default'  if self.\_is\_deprecated(self.\_item['title']):  heading\_format = 'deprecated'  column\_offset += self.\_write\_item\_columns(HEADING\_COLUMNS, heading\_format, column\_offset)  # Don't output leading columns here, because they depends on the format calculated for variation columns  column\_offset += len(LEADING\_COLUMNS)  column\_count, format = self.\_write\_item\_variations\_columns(column\_offset)  column\_offset += column\_count  # Now we are ready to fill leading columns!  self.\_write\_item\_columns(LEADING\_COLUMNS, format, len(CATEGORY\_COLUMNS + HEADING\_COLUMNS))  column\_offset += self.\_write\_item\_columns(VERIFICATION\_COLUMNS, format, column\_offset)  column\_offset += self.\_write\_item\_columns(PREV\_VERSION\_COLUNBS, format, column\_offset)  def \_write\_column(self, column, contents, format):  self.\_sheet.write\_cell(self.\_index, column, contents, format)  def \_write\_item\_columns(self, columns, format = 'default', column\_offset = 0):  for index, column in enumerate(columns):  label, width, key, override\_format = column  self.\_write\_column(column\_offset + index, self.\_get\_column\_value(key), override\_format or format)  return len(columns)  def \_get\_column\_value(self, key):  if key == 'category':  return self.\_category  elif key == 'index':  return int(self.\_item['index'])  elif key == 'title':  return self.\_item['title']  elif key == 'option\_index':  return int(self.\_option['option\_index'])  elif key == 'option\_title':  return self.\_option['option\_title']  elif key == 'template\_config':  return self.\_template\_curr\_config  elif key == 'verification\_chapter':  return self.\_verification\_chapter;  elif key == 'template\_prev\_config':  return self.\_template\_prev\_config;  elif key == 'applied\_prev\_config':  return self.\_prev\_config;  else:  return ''  def \_write\_item\_variations\_columns(self, column\_offset):  option\_id = self.\_option['option\_id']  column\_count = 0  row\_format = 'default'  verification\_chapter = ''  last\_conf = self.\_prev\_conf  last\_config = self.\_prev\_config  for version, conf in self.\_sheet.iterate\_all\_confs():  if version == ESR\_PREVIOUS.upper():  continue  config = self.\_get\_option\_config(conf)  format, status = self.\_determine\_format\_and\_status(conf, last\_conf, last\_config)  if last\_conf == self.\_prev\_conf:  row\_format = format  if option\_id in conf:  self.\_verification\_chapter = self.\_verification\_chapters.get(option\_id, '省略')  self.\_write\_column(column\_offset + column\_count, config, format)  self.\_write\_column(column\_offset + column\_count + 1, status, format)  column\_count += 2  last\_conf = conf  last\_config = config  return [column\_count, row\_format]  def \_determine\_format\_and\_status(self, conf, last\_conf, last\_config):  option = self.\_option  option\_id = option['option\_id']  status = ''  format = 'default'  config = self.\_get\_option\_config(conf)  modified = self.\_sanitize\_config(last\_config) != self.\_sanitize\_config(config)  if self.\_is\_deprecated(self.\_item['title']) or self.\_is\_deprecated(option['option\_title']):  format = 'deprecated'  elif option\_id in conf:  if option\_id not in last\_conf:  format, status = 'selected\_changed', '新規'  elif modified:  format, status = 'selected\_changed', '変更あり'  else:  format, status = 'selected', ''  elif last\_conf == self.\_prev\_conf:  if self.\_modified\_from\_prev\_version():  if self.\_added\_at\_this\_version():  format, status = 'changed', '新規（未設定）'  else:  format, status = 'changed', '変更あり（未設定）'  else:  if modified:  status = '削除'  return [format, status]  def \_modified\_from\_prev\_version(self):  return self.\_sanitize\_config(self.\_template\_curr\_config) != self.\_sanitize\_config(self.\_template\_prev\_config)  def \_added\_at\_this\_version(self):  return self.\_template\_prev\_config == ''  def \_is\_deprecated(self, string):  return '廃止' in string  def \_sanitize\_config(self, config):  return re.sub(' \*[^:]+:\n', '', config).strip()  def generate\_xlsx(workbook, confs, verification\_chapters, exclude\_worksheets):  formats = create\_formats(workbook)  prev\_conf = confs[ESR\_PREVIOUS.upper()]  for title, sources in WORKBOOKS:  if title in exclude\_worksheets:  continue  sheet = ConfigurationSheet(  confs,  formats,  workbook.add\_worksheet(title),  )  sheet.write\_header()  row\_index = 1  for source in sources:  # We always output items based on sources for the current version.  # In other words, the "current version" needs to define all deprecated/obsolete items  # if they still need to be visible in the output sheet.  base\_items = adlib.load(os.path.join(BASEDIR, ESR\_CURRENT, source))  prev\_items = adlib.load\_as\_dict(os.path.join(BASEDIR, ESR\_PREVIOUS, source))  sheet.merge\_category\_heading(row\_index, base\_items)  for item in base\_items:  sheet.try\_merge\_item\_heading(row\_index, item)  for option in item['options']:  row = ConfigurationRow(  sheet,  row\_index,  item,  option,  source,  prev\_conf,  prev\_items,  verification\_chapters,  )  row.write()  row\_index += 1  sheet.write\_legend(row\_index + 1)  def create\_formats(workbook):  def new\_format(\*\*kwargs):  return workbook.add\_format(dict(DEFAULT\_FORMAT, \*\*kwargs))  return {  'default': new\_format(),  'noborder': new\_format(border = 0),  'center': new\_format(align = 'center'),  'changed': new\_format(bold = True),  'deprecated': new\_format(bg\_color = '#dddddd'),  'question': new\_format(bg\_color = '#90ee90'),  'selected': new\_format(bg\_color = '#fffa95'),  'selected\_changed': new\_format(bg\_color = '#ffb571'),  }  #  # main  def load\_verification\_chapters(path):  try:  with open(path) as file:  return dict(csv.reader(file))  except FileNotFoundError:  return {}  def main(args):  confs = {}  outfile = 'config.xlsx'  exclude\_worksheets = []  opts, args = getopt.getopt(args, 'o:x:p:c:d:')  for key, value in opts:  if key == '-o':  outfile = value  elif key == '-x':  exclude\_worksheets = value.split(',')  elif key == '-p':  confs[ESR\_PREVIOUS.upper()] = value  elif key == '-c':  confs[ESR\_CURRENT.upper()] = value  elif key == '-d':  parts = value.split(':', 1)  confs[parts[0]] = parts[1]  verification\_chapters = {}  for arg in args:  if ESR\_PREVIOUS in arg and not ESR\_PREVIOUS.upper() in confs:  print('%s -> %s' % (ESR\_PREVIOUS, arg))  confs[ESR\_PREVIOUS.upper()] = arg  elif ESR\_CURRENT in arg and not ESR\_CURRENT.upper() in confs:  print('%s -> %s' % (ESR\_CURRENT, arg))  confs[ESR\_CURRENT.upper()] = arg  elif CHAPTERS\_CSV in arg:  print('Loading', os.path.basename(arg))  verification\_chapters = load\_verification\_chapters(arg)  for version, path in confs.items():  confs[version] = adlib.load\_as\_dict(path)  with xlsxwriter.Workbook(outfile) as workbook:  generate\_xlsx(workbook, confs, verification\_chapters, exclude\_worksheets)  print('Generated:', workbook.filename)  if \_\_name\_\_ == '\_\_main\_\_':  sys.exit(main(sys.argv[1:])) |