build-xlsx-1.py

```
#!/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',
    ]),
]
```

```
69
70      DEFAULT_FORMAT = {
71          'valign': 'top',
72          'border': 1,
73          'font_size': 8,
74          'font_name': 'MS Gothic',
75          'text_wrap': 1
76      }
77
78      #
79      # XLSX writer
80
81      def is_deprecated(x):
82          return '廃止' in x
83
84      def count_options(conf):
85          return sum(len(item['opts']) for item in conf)
86
87      def create_formats(wb):
88          def new_format(**kwargs):
89              return wb.add_format(dict(DEFAULT_FORMAT, **kwargs))
90          return {
91              'default':             new_format(),
92              'noborder':             new_format(border=0),
93              'center':               new_format(align='center'),
94              'deprecated':           new_format(bg_color='#dddddd'),
95              'question':             new_format(bg_color='#90ee90'),
96              'selected':            new_format(bg_color='#fffa95'),
97              'selected_changed': new_format(bg_color='#ffb571'),
98          }
99
100     def write_legend(sheet, formats, row):
101         sheet.write(row, 1, '', formats['selected'])
102         sheet.write(row, 2, '前バージョンから引き続き利用する項目', formats['noborder'])
103         sheet.write(row + 1, 1, '', formats['selected_changed'])
104         sheet.write(row + 1, 2, '前バージョンから異同がある項目', formats['noborder'])
105         sheet.write(row + 2, 1, '', formats['deprecated'])
106         sheet.write(row + 2, 2, '廃止済みの項目', formats['noborder'])
107
108     def write_header(sheet, formats):
109         curr = ESR_CURRENT.upper()
110         prev = ESR_PREVIOUS.upper()
111         fmt = formats['center']
112
113         sheet.freeze_panes(1, 0)
114
115         sheet.write(0, 0, 'カテゴリー', fmt)
116         sheet.write(0, 1, '項目設定番号', fmt)
117         sheet.write(0, 2, 'カスタマイズ項目（目的)', fmt)
118         sheet.write(0, 3, '状態', fmt)
119         sheet.write(0, 4, '選択肢番号', fmt)
120         sheet.write(0, 5, '選択肢', fmt)
121         sheet.write(0, 6, '設定内容の雛形¥n(%s)' % curr, fmt)
122         sheet.write(0, 7, '最終的に反映した設定値¥n(%s)' % curr, fmt)
123         sheet.write(0, 8, '%s→%s での変更' % (prev, curr), fmt)
124         sheet.write(0, 9, '検証手順書対応番号', fmt)
125         sheet.write(0, 11, '設定内容の雛形¥n(%s)' % prev, fmt)
126         sheet.write(0, 12, '最終的に反映した設定値¥n(%s)' % prev, fmt)
127
128         sheet.set_row(0, 25)
129         sheet.set_column(0, 12, None, formats['default'])
130         sheet.set_column(0, 0, 10)
131         sheet.set_column(1, 1, 10)
132         sheet.set_column(2, 2, 30)
133         sheet.set_column(3, 3, 5)
134         sheet.set_column(4, 4, 5)
135         sheet.set_column(5, 5, 20)
136         sheet.set_column(6, 6, 40)
137         sheet.set_column(7, 7, 40)
138         sheet.set_column(8, 8, 10)
139         sheet.set_column(9, 9, 10)
140         sheet.set_column(10, 10, 12)
```

```
141         sheet.set_column(11, 11, 40)
142         sheet.set_column(12, 12, 40)
143
144     def generate_xlsx(wb, conf_curr, conf_prev, chapters, excludes):
145         formats = create_formats(wb)
146
147         for title, files in WORKBOOK_DEF:
148             if title in excludes:
149                 continue
150
151             sheet = wb.add_worksheet(title)
152             write_header(sheet, formats)
153
154             row = 1
155             for fn in files:
156                 curr = adlib.load(os.path.join(BASEDIR, ESR_CURRENT, fn))
157                 prev = adlib.load_as_dict(os.path.join(BASEDIR, ESR_PREVIOUS, fn))
158                 sheet.merge_range(row, 0, row + count_options(curr) - 1, 0, '')
159
160                 for item in curr:
161                     if len(item['opts']) > 1:
162                         sheet.merge_range(row, 1, row + len(item['opts']) - 1, 1, '')
163                         sheet.merge_range(row, 2, row + len(item['opts']) - 1, 2, '')
164
165                     for opt in item['opts']:
166                         selected = ''
167                         status = ''
168                         chapter = ''
169                         fmt = formats['default']
170                         item_fmt = formats['default']
171                         opt_id = opt['opt_id']
172
173                         if is_deprecated(item['item_title']):
174                             item_fmt = formats['deprecated']
175                             fmt = formats['deprecated']
176                         elif is_deprecated(opt['opt_title']):
177                             fmt = formats['deprecated']
178                         elif opt_id in conf_curr:
179                             selected = 'y'
180                             chapter = chapters.get(opt_id, '省略')
181                             if opt_id not in conf_prev:
182                                 fmt, status = formats['selected_changed'], '新規'
183                             elif conf_prev[opt_id] != conf_curr[opt_id]:
184                                 fmt, status = formats['selected_changed'], '変更あり'
185                             else:
186                                 fmt, status = formats['selected'], ''
187
188                         sheet.write(row, 0, fn, formats['default'])
189                         sheet.write(row, 1, int(item['item_no']), item_fmt)
190                         sheet.write(row, 2, item['item_title'], item_fmt)
191                         sheet.write(row, 3, selected, fmt)
192                         sheet.write(row, 4, int(opt['opt_no']), fmt)
193                         sheet.write(row, 5, opt['opt_title'], fmt)
194                         sheet.write(row, 6, opt['conf'].strip(), fmt)
195                         sheet.write(row, 7, conf_curr.get(opt_id, ''), fmt)
196                         sheet.write(row, 8, status, fmt)
197                         sheet.write(row, 9, chapter, formats['default'])
198                         sheet.write(row, 10, '', formats['noborder'])
199                         sheet.write(row, 11, prev.get(opt_id, ''), fmt)
200                         sheet.write(row, 12, conf_prev.get(opt_id, ''), fmt)
201                         row += 1
202             write_legend(sheet, formats, row+1)
203
204     #
205     # main
206
207     def load_chapters(path):
208         try:
209             with open(path) as fp:
210                 return dict(csv.reader(fp))
211         except FileNotFoundError:
212             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

```
#!/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

```

```
141         sheet.write(0, col_count+1, '検証手順書対応番号', fmt)
142         sheet.write(0, col_count+3, '設定内容の雛形¥n(%s)' % ESR_PREVIOUS.upper(), fmt)
143         sheet.write(0, col_count+4, '反映した設定値¥n(%s)' % ESR_PREVIOUS.upper(), fmt)
144
145         sheet.set_row(0, 25)
146         sheet.set_column(0, 0, 10)
147         sheet.set_column(1, 1, 10)
148         sheet.set_column(2, 2, 30)
149         sheet.set_column(3, 3, 5)
150         sheet.set_column(4, 4, 20)
151         sheet.set_column(5, 5, 40)
152
153         sheet.set_column(col_count+1, col_count+1, 10)
154         sheet.set_column(col_count+2, col_count+2, 12)
155         sheet.set_column(col_count+3, col_count+3, 40)
156         sheet.set_column(col_count+4, col_count+4, 40)
157         col_count+=4
158
159         sheet.set_column(0, col_count, None, formats['default'])
160
161     def generate_xlsx(wb, conf, chapters, excludes):
162         formats = create_formats(wb)
163         prev_conf = conf[ESR_PREVIOUS.upper()]
164
165         for title, files in WORKBOOK_DEF:
166             if title in excludes:
167                 continue
168
169             sheet = wb.add_worksheet(title)
170             write_header(sheet, formats, conf)
171
172             row = 1
173             for fn in files:
174                 curr = adlib.load(os.path.join(BASEDIR, ESR_CURRENT, fn))
175                 prev = adlib.load_as_dict(os.path.join(BASEDIR, ESR_PREVIOUS, fn))
176                 sheet.merge_range(row, 0, row + count_options(curr) - 1, 0, '')
177
178                 for item in curr:
179                     if len(item['opts']) > 1:
180                         sheet.merge_range(row, 1, row + len(item['opts']) - 1, 1, '')
181                         sheet.merge_range(row, 2, row + len(item['opts']) - 1, 2, '')
182
183                     for opt in item['opts']:
184                         status = ''
185                         chapter = ''
186                         fmt = formats['default']
187                         item_fmt = formats['default']
188                         opt_id = opt['opt_id']
189
190                         applied_prev_conf = prev_conf.get(opt_id, {'conf':''})['conf']
191                         template_curr_conf = opt['conf'].strip()
192                         template_prev_conf = prev.get(opt_id, {'conf':''})['conf']
193
194                         if is_deprecated(item['item_title']):
195                             item_fmt = formats['deprecated']
196
197                         col_count = 5
198                         base_conf = prev_conf
199                         applied_base_conf = applied_prev_conf
200                         for key, variation_conf in conf.items():
201                             if key == key == ESR_PREVIOUS.upper():
202                                 continue
203
204                             variation_status = ''
205                             variation_fmt = ''
206                             applied_variation_conf = variation_conf.get(opt_id, {'conf':''})['conf']
207
208                             if is_deprecated(item['item_title']) or is_deprecated(opt['opt_title']):
209                                 variation_fmt = formats['deprecated']
210                             elif opt_id in variation_conf:
211                                 chapter = chapters.get(opt_id, '省略')
212                                 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

```
#!/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):
```

```
141             return self._confs.items()
142
143         def write_cell(self, row, column, contents, format):
144             self._sheet.write(row, column, contents, self._formats[format])
145
146         def _set_cell_visual(self, row, column, width, format = None):
147             if format:
148                 self._sheet.set_column(row, column, width, self._formats[format])
149             else:
150                 self._sheet.set_column(row, column, width)
151
152         def write_header(self):
153             sheet = self._sheet
154
155             sheet.freeze_panes(1, 0)
156             sheet.set_row(0, 25)
157
158             column_offset = 0
159             column_offset += self._write_header_columns(CATEGORY_COLUMNS, 0)
160             column_offset += self._write_header_columns(HEADING_COLUMNS, column_offset)
161             column_offset += self._write_header_columns(LEADING_COLUMNS, column_offset)
162
163             last_variation = ESR_PREVIOUS.upper()
164             for variation in self._confs.keys():
165                 if variation == ESR_PREVIOUS.upper():
166                     continue
167                 columns = variation_columns(variation, last_variation)
168                 column_offset += self._write_header_columns(columns, column_offset)
169                 last_variation = variation
170
171             column_offset += self._write_header_columns(VERIFICATION_COLUMNS, column_offset)
172             column_offset += self._write_header_columns(PREV_VERSION_COLUNBS, column_offset)
173
174         def _write_header_columns(self, columns, column_offset):
175             for index, column in enumerate(columns):
176                 label, width, _key, _format = column
177                 self.write_cell(0, column_offset + index, label, 'center')
178                 self._set_cell_visual(column_offset + index, column_offset + index, width)
179             return len(columns)
180
181         def merge_category_heading(self, row, items):
182             for index, _column in enumerate(CATEGORY_COLUMNS):
183                 self._sheet.merge_range(row, index, row + self._count_options(items) - 1, index, '')
184
185         def _count_options(self, items):
186             return sum(len(item['options']) for item in items)
187
188         def try_merge_item_heading(self, row, item):
189             if len(item['options']) <= 1:
190                 return
191             sheet = self._sheet
192             column_offset = len(CATEGORY_COLUMNS)
193             for index, _column in enumerate(HEADING_COLUMNS):
194                 sheet.merge_range(row, column_offset + index, row + len(item['options']) - 1, column_offset + index, '')
195
196         def write_legend(self, row):
197             self.write_cell(row,     1, '',                                   'selected')
198             self.write_cell(row,     2, '前バージョンから引き続き利用する項目', 'noborder')
199             self.write_cell(row + 1, 1, '',                                   'selected_changed')
200             self.write_cell(row + 1, 2, '前バージョンから異同がある項目',       'noborder')
201             self.write_cell(row + 2, 1, '',                                   'deprecated')
202             self.write_cell(row + 2, 2, '廃止済みの項目',                      'noborder')
203
204     class ConfigurationRow:
205
206         def __init__(self, sheet, index, item, option, category,
207                      prev_conf, prev_items, verification_chapters):
208             self._sheet                 = sheet
209             self._index                 = index
210             self._item                  = item
211             self._option                = option
212             self._category              = category
```

```
213             self._prev_conf                 = prev_conf
214             self._verification_chapters = verification_chapters
215             self._verification_chapter    = ''
216
217             self._prev_config               = self._get_option_config(self._prev_conf)
218             self._template_prev_config    = self._get_option_config(prev_items)
219             self._template_curr_config    = option['config'].strip()
220
221         def _get_option_config(self, conf_or_items):
222             found_option = conf_or_items.get(self._option['option_id'])
223             if not found_option:
224                 return ''
225             return found_option['config']
226
227         def write(self):
228             column_offset = 0
229             column_offset += self._write_item_columns(CATEGORY_COLUMNS)
230
231             # Heading column must be written for all rows, otherwise merged cells will have
232             # a partial border line just for the first row.
233             heading_format = 'default'
234             if self._is_deprecated(self._item['title']):
235                 heading_format = 'deprecated'
236             column_offset += self._write_item_columns(HEADING_COLUMNS, heading_format, column_offset)
237
238             # Don't output leading columns here, because they depends on the format calculated for variation columns
239             column_offset += len(LEADING_COLUMNS)
240             column_count, format = self._write_item_variations_columns(column_offset)
241             column_offset += column_count
242
243             # Now we are ready to fill leading columns!
244             self._write_item_columns(LEADING_COLUMNS, format, len(CATEGORY_COLUMNS + HEADING_COLUMNS))
245
246             column_offset += self._write_item_columns(VERIFICATION_COLUMNS, format, column_offset)
247             column_offset += self._write_item_columns(PREV_VERSION_COLUNBS, format, column_offset)
248
249         def _write_column(self, column, contents, format):
250             self._sheet.write_cell(self._index, column, contents, format)
251
252         def _write_item_columns(self, columns, format = 'default', column_offset = 0):
253             for index, column in enumerate(columns):
254                 label, width, key, override_format = column
255                 self._write_column(column_offset + index, self._get_column_value(key), override_format or format)
256             return len(columns)
257
258         def _get_column_value(self, key):
259             if key == 'category':
260                 return self._category
261             elif key == 'index':
262                 return int(self._item['index'])
263             elif key == 'title':
264                 return self._item['title']
265             elif key == 'option_index':
266                 return int(self._option['option_index'])
267             elif key == 'option_title':
268                 return self._option['option_title']
269             elif key == 'template_config':
270                 return self._template_curr_config
271             elif key == 'verification_chapter':
272                 return self._verification_chapter;
273             elif key == 'template_prev_config':
274                 return self._template_prev_config;
275             elif key == 'applied_prev_config':
276                 return self._prev_config;
277             else:
278                 return ''
279
280         def _write_item_variations_columns(self, column_offset):
281             option_id = self._option['option_id']
282
283             column_count                = 0
284             row_format                  = 'default'
```

```
285             verification_chapter = ''
286
287             last_conf     = self._prev_conf
288             last_config = self._prev_config
289             for version, conf in self._sheet.iterate_all_confs():
290                 if version == ESR_PREVIOUS.upper():
291                     continue
292
293                 config                = self._get_option_config(conf)
294                 format, status = self._determine_format_and_status(conf, last_conf, last_config)
295
296                 if last_conf == self._prev_conf:
297                    row_format = format
298
299                 if option_id in conf:
300                     self._verification_chapter = self._verification_chapters.get(option_id, '省略')
301
302                 self._write_column(column_offset + column_count,         config, format)
303                 self._write_column(column_offset + column_count + 1, status, format)
304
305                 column_count += 2
306                 last_conf     = conf
307                 last_config = config
308
309             return [column_count, row_format]
310
311         def _determine_format_and_status(self, conf, last_conf, last_config):
312             option       = self._option
313             option_id = option['option_id']
314
315             status     = ''
316             format     = 'default'
317             config     = self._get_option_config(conf)
318             modified = self._sanitize_config(last_config) != self._sanitize_config(config)
319
320             if self._is_deprecated(self._item['title']) or self._is_deprecated(option['option_title']):
321                 format = 'deprecated'
322             elif option_id in conf:
323                 if option_id not in last_conf:
324                     format, status = 'selected_changed', '新規'
325                 elif modified:
326                     format, status = 'selected_changed', '変更あり'
327                 else:
328                     format, status = 'selected', ''
329             elif last_conf == self._prev_conf:
330                 if self._modified_from_prev_version():
331                    if self._added_at_this_version():
332                         format, status = 'changed', '新規（未設定）'
333                    else:
334                         format, status = 'changed', '変更あり（未設定）'
335             else:
336                 if modified:
337                     status = '削除'
338
339             return [format, status]
340
341         def _modified_from_prev_version(self):
342             return self._sanitize_config(self._template_curr_config) != self._sanitize_config(self._template_prev_config)
343
344         def _added_at_this_version(self):
345             return self._template_prev_config == ''
346
347         def _is_deprecated(self, string):
348             return '廃止' in string
349
350         def _sanitize_config(self, config):
351             return re.sub('*[^:]+:¥n', '', config).strip()
352
353     def generate_xlsx(workbook, confs, verification_chapters, exclude_worksheets):
354         formats = create_formats(workbook)
355         prev_conf = confs[ESR_PREVIOUS.upper()]
356
```

```
    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:]))
```