ddrsplit.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function

import sys
import os

import io
import unicodedata
import time
import binascii
import base64
import hashlib

import pprint
pp = pprint.pprint

import pdb

import xml.etree.cElementTree
ElementTree = xml.etree.cElementTree

import xml.parsers.expat

import Config

import ReferenceCollector

# py3 stuff

py3 = False
try:
    unicode('')
    punicode = unicode
    pstr = str
    punichr = unichr
except NameError:
    punicode = str
    pstr = bytes
    py3 = True
    punichr = chr

#
# globals
#
gCancel = False
gREF = ReferenceCollector.ReferenceCollector()

#
# tools
#
def makeunicode(s, srcencoding="utf-8", normalizer="NFC"):
    if type(s) not in (punicode, pstr):
        s = str( s )
    if type(s) != punicode:
        s = punicode(s, srcencoding)
    s = unicodedata.normalize(normalizer, s)
    return s

def stringhash( s ):
    m = hashlib.sha1()
    m.update(s)
    return m.hexdigest().upper()

def logfunction(s):
    s = s + u"\n"
    sys.stdout.write( s.encode("utf-8") )

#
# parsers
#
def xmlexportfolder(basefolder, dbname, category, obname, obid="", ext=".xml"):
    # create or get folder where to put layout, script or basetable xml
    path = os.path.abspath(basefolder)

    catfolder = os.path.join( path, dbname, category)

    if not os.path.exists(catfolder):
        os.makedirs( catfolder )

    if obid:
        obid = str(obid).rjust(7,"0") + " "
    filename = obid + obname + ext
    filename = filename.replace('/', '_')
    filename = filename.replace(':', '_')
    filename = filename.replace('\\', '_')

    fullpath = os.path.join( catfolder, filename)
    fullpath = makeunicode( fullpath, normalizer="NFD" )
    return fullpath

def get_text_object(cfg, cur_fmpxml, cur_db, cur_fmpbasename, cur_node, cur_object):
    # check for global variables and merge fields
    pass

def get_script_step(cfg, cur_fmpxml, cur_db, cur_fmpbasename, cur_node, cur_object):

    # BAUSTELLE

    # check for scripstep and field parameters
    # catch
    #       perform scrip
    #       exit script (parameter)
    #       set variable
    #       install on  timer script
    #       go to layout
    #       go to related record
    #       go to object
    #       go to field
    #       enter find mode
    step_id = cur_node.attrib.get("id", -1)
    step_name = cur_node.attrib.get("name", "NO SCRIPTSTEP NAME")

    fref_id = fref_name = step_calc_text = None

    for subnode in cur_node.iter():
        if subnode.tag == "FileReference":
            fref_id = subnode.attrib.get("id", -1)
            fref_name = subnode.attrib.get("name", "NO FILEREFERENCE NAME")

        elif subnode.tag == "DisplayCalculation":
            get_displaycalculation(cfg, cur_fmpxml, cur_db, cur_fmpbasename, subnode)
        elif subnode.tag == "Calculation":
            step_calc_text = subnode.text
    external = fref_id and fref_name

def get_displaycalculation(cfg, cur_fmpxml, cur_db, cur_fmpbasename, cur_node):
    clc_text = clc_noref = clc_fnctref = clc_fieldref = clc_cf = ""

    for node in cur_node.iter():
        dpc_tag = node.tag
        dpc_typ = node.attrib.get( "type", "")
        if dpc_typ == "NoRef":
            clc_noref = node.text
        elif dpc_typ == "FunctionRef":
            pass
        elif dpc_typ == "FieldRef":
            # <Field id="1" name="F1" table="to_Test1" />
            pass
        elif dpc_typ == "CustomFunctionRef":
            pass

def get_authfilecatalog(cfg, cur_fmpxml, cur_db, cur_fmpbasename, authfiles,
                        groups, exportfolder, idx):

    for authfile in authfiles:
        authfile_attr = authfile.attrib
        authfile_tag = authfile.tag
        authfile_name = authfile_attr.get("name", "NONAME")

        cur_object = (cur_fmpxml, 'AuthFile', authfile_name)

        path = "AuthFiles"

        s = ElementTree.tostring(authfile, encoding="utf-8", method="xml")

        sortid = authfile_attr.get("id", "0").rjust(7,"0")

        objectID = sortid
        if cfg.ignoreFilenameIDs:
            objectID = ""
        path = xmlexportfolder(exportfolder,
                               cur_fmpbasename,
                               path,
                               authfile_name,
                               objectID)
        f = open(path, "wb")
        f.write( s )
        f.close()
        idx += 1
    return idx

def get_externaldatasources(cfg, cur_fmpxml, cur_db, cur_fmpbasename, externaldatasources,
                            groups, exportfolder, idx):

    for externaldatasource in externaldatasources:
        externaldatasource_attr = externaldatasource.attrib
        externaldatasource_tag = externaldatasource.tag
        externaldatasource_name = externaldatasource_attr.get("name", "NONAME")

        cur_object = (cur_fmpxml, 'ExternalDataSource', externaldatasource_name)

        path = "ExternalDataSources"

        s = ElementTree.tostring(externaldatasource, encoding="utf-8", method="xml")

        sortid = externaldatasource_attr.get("id", "0").rjust(7,"0")

        objectID = sortid
        if cfg.ignoreFilenameIDs:
            objectID = ""
        path = xmlexportfolder(exportfolder,
                               cur_fmpbasename,
                               path,
                               externaldatasource_name,
                               objectID)
        f = open(path, "wb")
        f.write( s )
        f.close()
        idx += 1
    return idx

def get_themecatalog(cfg, cur_fmpxml, cur_db, cur_fmpbasename, themes,
                            groups, exportfolder, idx):

    for theme in themes:
        theme_attr = theme.attrib
        theme_tag = theme.tag
        theme_name = theme_attr.get("name", "NONAME")

        cur_object = (cur_fmpxml, 'ThemeCatalog', theme_name)

        path = "Themes"

        s = ElementTree.tostring(theme, encoding="utf-8", method="xml")

        sortid = theme_attr.get("id", "0").rjust(7,"0")

        objectID = sortid
        if cfg.ignoreFilenameIDs:
            objectID = ""
        path = xmlexportfolder(exportfolder,
                               cur_fmpbasename,
                               path,
                               theme_name,
                               objectID)
        f = open(path, "wb")
        f.write( s )
        f.close()
        idx += 1
    return idx

def get_basedirectories(cfg, cur_fmpxml, cur_db, cur_fmpbasename, basedirectories,
                            groups, exportfolder, idx):

    for basedirectory in basedirectories:
        basedirectory_attr = basedirectory.attrib
        basedirectory_tag = basedirectory.tag
        basedirectory_name = basedirectory_attr.get("name", "NONAME")

        cur_object = (cur_fmpxml, 'BaseDirectoryCatalog', basedirectory_name)

        path = "BaseDirectoryCatalog"

        s = ElementTree.tostring(basedirectory, encoding="utf-8", method="xml")

        sortid = basedirectory_attr.get("id", "0").rjust(7,"0")

        objectID = sortid
        if cfg.ignoreFilenameIDs:
            objectID = ""
        path = xmlexportfolder(exportfolder,
                               cur_fmpbasename,
                               path,
                               basedirectory_name,
                               objectID)
        f = open(path, "wb")
        f.write( s )
        f.close()
        idx += 1
    return idx

def get_layouts_and_groups(cfg, cur_fmpxml, cur_db, cur_fmpbasename, laynode,
                           groups, exportfolder, idx):

    for layout in laynode:
        layout_attr = layout.attrib
        layout_tag = layout.tag
        layout_name = layout_attr.get("name", "NONAME")

        cur_object = (cur_fmpxml, 'Layout', layout_name)

        if layout_tag == "Group":
            grp_attrib = layout_attr
            groupid = layout_attr.get("id", "0")

            # get layout folder name
            groupname = (  groupid.rjust(7,"0")
                         + ' '
                         + layout_name )

            if cfg.layoutOrder:
                groupname = (  str(idx).rjust(5,"0")
                             + ' '
                             + groupid.rjust(7,"0")
                             + ' '
                             + layout_name )

            if cfg.ignoreFilenameIDs:
                groupname = layout_name

            groups.append( groupname )

            idx += 1
            idx = get_layouts_and_groups(cfg, cur_fmpxml, cur_db, cur_fmpbasename, layout,
                                         groups, exportfolder, idx)
            groups.pop()
        else:
            path = "Layouts"

            if groups and cfg.layoutGroups:
                path = os.path.join("Layouts", *groups)
            s = ElementTree.tostring(layout, encoding="utf-8", method="xml")

            sortid = layout_attr.get("id", "0").rjust(7,"0")
            if cfg.layoutOrder:
                sortid = (str(idx).rjust(5,"0")
                          + ' '
                          + layout_attr.get("id", "0").rjust(7,"0") )

            objectID = sortid
            if cfg.ignoreFilenameIDs:
                objectID = ""
            path = xmlexportfolder(exportfolder,
                                   cur_fmpbasename,
                                   path,
                                   layout_name,
                                   objectID)
            f = open(path, "wb")
            f.write( s )
            f.close()
            idx += 1

            if not cfg.assets:
                continue

            for l in layout.getchildren():
                t = l.tag
                if t == u'Object':
                    get_layout_object(cfg, cur_fmpxml, cur_db, cur_fmpbasename,
                                      l, cur_object, exportfolder)
    return idx

def get_layout_object(cfg, cur_fmpxml, cur_db, cur_fmpbasename, laynode,
                      cur_object, exportfolder):
    nodes = list(laynode)
    extensions = dict(zip( ("JPEG","PDF ", "PNGf", "PICT",
                            "GIFf", "8BPS", "BMPf"),
                           (".jpg",".pdf", ".png", ".pict",
                            ".gif", ".psd", ".bmp")))
    exttypelist = extensions.keys()

    cur_tableOccurrenceName = cur_tableOccurrenceID = ""

    for node in nodes:
        cur_tag = node.tag

        if cur_tag == u'Object':
            # get layout object
            get_layout_object(cfg, cur_fmpxml, cur_db, cur_fmpbasename, node,
                              cur_object, exportfolder)

        elif cur_tag == u'ObjectStyle':
            continue

        elif cur_tag == u'Table':
            # <Table id="13631489" name="to_Bildarchiv" />
            cur_tableOccurrenceID = node.get("id", -1)
            cur_tableOccurrenceName = node.get("name",
                                        "NO TABLE OCCURRENCE NAME FOR LAYOUT")
            cur_objectID = gREF.addObject( cur_object )
            gREF.addFilemakerAttribute( cur_objectID, "tableOccurrenceID",
                                                       cur_tableOccurrenceID)
            gREF.addFilemakerAttribute( cur_objectID, "tableOccurrenceName",
                                                       cur_tableOccurrenceName)

        elif cur_tag == u'GraphicObj':
            for grobnode in node:
                if grobnode.tag == "Stream":
                    stype = []
                    sdata = ""
                    for streamnode in grobnode:
                        streamtag = streamnode.tag
                        streamtext = streamnode.text
                        if streamtag == "Type":
                            if streamtext not in exttypelist:
                                stype.append( '.' + streamtext )
                            else:
                                stype.append( streamtext )
                        elif streamtag in ("Data", "HexData"):
                            if not stype:
                                continue
                            curtype = stype[-1]
                            ext = extensions.get( curtype, False )
                            if not ext:
                                ext = curtype
                            data = None
                            if streamtag == "HexData":
                                try:
                                    data = binascii.unhexlify ( streamtext )
                                except TypeError as err:
                                    pass
                            elif streamtag == "Data":
                                try:
                                    data = base64.b64decode( streamtext )
                                except TypeError as err:
                                    pass
                            if not data:
                                continue

                            fn = stringhash( data )
                            path = xmlexportfolder(exportfolder,
                                                   cur_fmpbasename,
                                                   "Assets",
                                                   fn,
                                                   "",
                                                   ext)
                            # write Asset file
                            if not os.path.exists( path ):
                                f = open(path, "wb")
                                f.write( data )
                                f.close()

        # the following tags are for reference collection only

        # laynode is current node
        # cur_object is ref1
        elif cur_tag == u'GroupButtonObj':
            # recurse
            get_layout_object(cfg, cur_fmpxml, cur_db, cur_fmpbasename,
                              node, cur_object, exportfolder)

        elif cur_tag == u'FieldObj':
            # check for scripstep and field parameters
            for subnode in node.iter():
                if subnode.tag == "Field":
                    # <Field id="2" maxRepetition="1" name="F2"
                    # repetition="1" table="Test1" />
                    fld_id = int(subnode.attrib.get("id", -1))
                    fld_name = subnode.attrib.get("name",
                                                    "NO FIELD NAME")
                    fld_to = subnode.attrib.get("table",
                                                    "NO TABLE OCCURRENCE")

                    fld_obj = (cur_object[0], "Field", fld_name, fld_to)
                    fld_obj_id = gREF.addObject( fld_obj )
                    gREF.addFilemakerAttribute(fld_obj_id, "id", fld_id)

                    gREF.addReference(cur_object, fld_obj)

        elif cur_tag == u'Step':
            get_script_step(cfg, cur_fmpxml, cur_db, cur_fmpbasename,
                            node, cur_object)
        elif cur_tag == u'TextObj':
            get_text_object(cfg, cur_fmpxml, cur_db, cur_fmpbasename,
                                node, cur_object)

def get_scripts_and_groups(cfg, cur_fmpxml, cur_db, cur_fmpbasename, scriptnode,
                           exportfolder, groups, namecache, idx):

    for scpt in scriptnode:
        if scpt.tag == "Script":
            path = "Scripts"
            if groups and cfg.scriptGroups:
                path = os.path.join("Scripts", *groups)

            sortid = scpt.get("id", "0").rjust(7,"0")
            if cfg.scriptOrder:
                sortid = (str(idx).rjust(5,"0")
                          + ' '
                          + scpt.get("id", "0").rjust(7,"0") )
            s = ElementTree.tostring(scpt, encoding="utf-8", method="xml")

            objectID = sortid
            if cfg.ignoreFilenameIDs:
                objectID = ""
            path = xmlexportfolder(exportfolder, cur_fmpbasename, path,
                                   scpt.get("name", "NONAME"),
                                   objectID)
            idx += 1
            f = open(path, "wb")
            f.write( s )
            f.close()

        elif scpt.tag == "Group":
            grp_attrib = scpt.attrib
            groupid = grp_attrib.get("id", "0")

            # script folder name (if any)
            groupname = (groupid.rjust(7,"0")
                         + ' '
                         + grp_attrib.get("name", "No folder name") )

            if cfg.scriptOrder:
                groupname = (str(idx).rjust(5,"0")
                             + ' ' + groupid.rjust(7,"0")
                             + ' ' + grp_attrib.get("name", "No folder name"))

            if cfg.ignoreFilenameIDs:
                groupname = grp_attrib.get("name", "No folder name")
            groups.append( groupname )

            idx += 1
            idx = get_scripts_and_groups(cfg, cur_fmpxml, cur_db, cur_fmpbasename,
                                         scpt, exportfolder, groups, namecache, idx)
            groups.pop()
    return idx

def get_relationshipgraph_catalog(cfg, cur_fmpxml, cur_db, cur_fmpbasename,
                                  rg_cat, exportfolder):
    for tablst in rg_cat:
        if tablst.tag == u'TableList':
            for tab in tablst:
                if tab.tag == u'Table':

                    to_attr = tab.attrib
                    to_name = tab.attrib.get("name", "NO TABLE OCCURRENCE NAME")
                    to_id = tab.attrib.get("id", -1)
                    to_btid = tab.attrib.get("baseTableId", -1)
                    to_bt = tab.attrib.get("baseTable", "NO BASETABLE FOR TABLE OCCURRENCE")

                    s = ElementTree.tostring(tab, encoding="utf-8", method="xml")

                    objectID = to_id
                    if cfg.ignoreFilenameIDs:
                        objectID = ""
                    path = xmlexportfolder(exportfolder,
                                           cur_fmpbasename,
                                           "Relationships/TableList",
                                           to_name,
                                           objectID)
                    f = open(path, "wb")
                    f.write( s )
                    f.close()

                    external = eto_id = eto_name = False
                    for node in tab.iter():
                        if node.tag == "FileReference":
                            external = True
                            eto_id = node.get("id", -1)
                            eto_name = node.get("name",
                                                "NO EXTERNAL FILEREF NAME FOR TABLE OCCURRENCE")

                    toObject = (cur_fmpxml, "TableOccurrence", to_name)
                    if external:
                        toObject = (cur_fmpxml, "ExternalTableOccurrence", to_name)

                    toObjectId = gREF.addObject( toObject )
                    gREF.addFilemakerAttribute( toObjectId, 'baseTableId', to_btid)
                    gREF.addFilemakerAttribute( toObjectId, 'baseTable', to_bt)
                    if external:
                        gREF.addFilemakerAttribute( toObjectId, 'fileReferenceID', eto_id)
                        gREF.addFilemakerAttribute( toObjectId, 'fileReferenceName', eto_name)

        # <Table baseTable="bt_Bildarchiv" baseTableId="32769"
        #  color="#777777" id="13631489" name="to_Bildarchiv" />

        # <Table baseTable="bt_Text" baseTableId="32769" color="#777777"
        # id="13631498" name="eto_TEX_artnum">
        # <FileReference id="1" name="Text" />
        # </Table>

            elif tablst.tag == u'RelationshipList':
                for rel in tablst:
                    if rel.tag == u'Relationship':
                        rel_cat = {}
                        re_attr = rel.attrib
                        relid = re_attr.get("id", "0")
                        rel_cat['id'] = re_attr.get("id", "0")

                        for rel_component in rel.getchildren():
                            if rel_component.tag == "LeftTable":
                                rel_cat['lefttable'] = rel_component.attrib.get("name",
                                                                    "NO-LEFTTABLENAME")
                            elif rel_component.tag == "RightTable":
                                rel_cat['righttable'] = rel_component.attrib.get("name",
                                                                    "NO-LEFTTABLENAME")

                        s = ElementTree.tostring(rel, encoding="utf-8", method="xml")
                        filename = (rel_cat['lefttable']
                                    + "---"
                                    + rel_cat['righttable'])

                        objectID = rel_cat['id']
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "Relationships/Relationship",
                                               filename,
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()

    def main(cfg):

        xmlfile = cfg.summaryfile
        xml_folder, xmlfilename = os.path.split( xmlfile )

        ddr = ElementTree.parse( xmlfile )

        summary = ddr.getroot()

        files = summary.findall( "File" )
        nooffiles = len( files )

        filelist = {}

        starttime = time.time()

        log = logfunction
        if cfg.logfunction:
            log = cfg.logfunction

        for fmpreport in summary.getiterator("FMPReport"):
            for xmlfile in fmpreport.getiterator("File"):
                xml_fmpfilename = xmlfile.get("name", "NO FILE NAME")
                xml_xmllink = xmlfile.get("link", "")
                xml_fmppath = xmlfile.get("path", "")

                if not xml_xmllink:
                    s = u"\nERROR: Could not find XML file '%s'\nContinue.\n"
                    log( s %  xml_xmllink)
                    continue

                # xml_xmllink
                # cleanup filename
                while xml_xmllink.startswith( './/' ): xml_xmllink = xml_xmllink[ 3: ]
                while xml_xmllink.startswith( './' ):  xml_xmllink = xml_xmllink[ 2: ]

                xmlbasename, ext  = os.path.splitext( xml_xmllink )

                filelist[ xml_xmllink ] = (xml_fmpfilename, xml_fmppath, xmlbasename)

        for cur_xml_file_name in filelist.keys():

            # path to DDR-XML file
            next_xml_file_path = os.path.join( xml_folder, cur_xml_file_name )

            # some UI glitz
            line = '-' * 100
            log( u"\n\n%s\n\nXMLFILE: %s" % (line, cur_xml_file_name) )
            print( "filelist[ xml_xmllink ]:", repr(filelist[ cur_xml_file_name ]) )
            # parse xml file
            try:
                basenode = ElementTree.parse( next_xml_file_path )
            except  (xml.parsers.expat.ExpatError, SyntaxError) as v:
                xml.parsers.expat.error()
                log( u"EXCEPTION: '%s'" % v )
                log( u"Failed parsing '%s'\n" % next_xml_file_path )
                continue

            # more often the xml filename is required for identification
            cur_db = filelist[ cur_xml_file_name ][0]
            cur_fmpbasename = filelist[ cur_xml_file_name ][2]
            cur_fmpxml = cur_xml_file_name

            cur_fileRef = (cur_fmpxml, "DatabaseFile", cur_db)

            exportfolder = cfg.exportfolder

            # relationships need to be analyzed first for the baseTable -> TO graph
            # for that to happen, filereferences must go before that

            print
            print

            #
            # FileReferenceCatalog
            #
            # todo check if refs && cfg.filereferences
            if 1: #cfg.filereferences:
                log( u'File References "%s"' % cur_fmpxml )
                for fr_cat in basenode.getiterator ( "FileReferenceCatalog" ):
                    for fileref in fr_cat.getchildren():
                        fileref_attrib = fileref.attrib
                        prefix = ""
                        if fileref.tag == "OdbcDataSource":
                            prefix = "ODBC-"
                        elif fileref.tag == "FileReference":
                            prefix = "FREF-"
                            #
                            # <FileReference id="2" link="Menu_fp7.xml" name="Menu"
                            #                pathList="file:Menu.fp7" />
                            #
                            frf_id = fileref.attrib.get("id", -1)
                            frf_link = fileref.attrib.get("link", "NO DDR.XML FILE")
                            frf_name = fileref.attrib.get("name", "NO FILEREF NAME")
                            frf_pathList = fileref.attrib.get("pathList",
                                                                    "NO FILEREF PATHLIST")
                            gREF.addFileReference(cur_xml_file_name, frf_link, frf_name,
                                                                    frf_id, frf_pathList)

                            frf_object = (cur_xml_file_name, 'FileReference', frf_name)
                            gREF.addObject(frf_object)
                            gREF.addReference(cur_fileRef, frf_object)

                        else:
                            prefix = "UNKN-"
                        name = prefix + fileref_attrib.get("name", "NONAME")

                        s = ElementTree.tostring(fileref,
                                                 encoding="utf-8",
                                                 method="xml")

                        objectID = fileref_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "Filereferences",
                                               name,
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists,
                # merge fields, scripts, TOs, FileReferences

            #
            # relationship graph
            #
            if cfg.relationships:
                log( u'Relationship Graph "%s"' % cur_fmpxml )
                for rg_cat in basenode.getiterator ( "RelationshipGraph" ):
                    get_relationshipgraph_catalog(cfg, cur_fmpxml, cur_db,
                                                  cur_fmpbasename, rg_cat, exportfolder)
                # collect references from FRF to FRF

            #
            # base table catalog
            #
            if cfg.basetables:
                log( u'Base Tables "%s"' % cur_fmpxml )
                for base_table_catalog in basenode.getiterator( u'BaseTableCatalog' ):
                    for base_table in base_table_catalog.getiterator( u'BaseTable' ):
                        bt_name = base_table.get("name", "NONAME")
                        bt_id = base_table.get("id", "0")
                        s = ElementTree.tostring(base_table,
                                                 encoding="utf-8",
                                                 method="xml")

                        objectID = bt_id
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "Basetables",
                                               bt_name,
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()

                        cur_btRef = (cur_fmpxml, "BaseTable", bt_name)
                        bt_objID = gREF.addObject( cur_btRef )

                        # make the basetable id known without using it for references
                        gREF.addFilemakerAttribute(bt_objID, "id", bt_id)

                        gREF.addReference( cur_fileRef, cur_btRef)

                        # TODO
                        #
                        # FIELDS
                        #
                        # cur_db, cur_btRef, bt_name, bt_id, bt_objID
                        for field_catalog in base_table.getiterator( u'FieldCatalog' ):
                            for field in field_catalog.getiterator( u'Field' ):
                                # dataType="Date"
                                # fieldType="Normal"
                                # id="9"
                                # name="dat_BAR_created"
                                fld_name = field.get("name", "NONAME")
                                fld_id = field.get("id", "0")
                                fld_type = field.get("fieldType", "NO FIELD TYPE")
                                fld_dataType = field.get("dataType", "NO DATA TYPE")

                                cur_fldRef = (cur_fmpxml, "Field", fld_name, bt_name)
                                fld_objID = gREF.addObject( cur_fldRef )

                                gREF.addFilemakerAttribute(fld_objID, "id", fld_id)
                                gREF.addFilemakerAttribute(fld_objID, "dataType",
                                                                      fld_dataType)
                                gREF.addFilemakerAttribute(fld_objID, "fieldType",
                                                                      fld_id)

                                gREF.addReference( cur_btRef, cur_fldRef)
                                # TODO
                                #
                                # add ref to TO (needs Calculations)

                # collect references to fields, CFs, value lists, TOs, FileReferences

            #
            # LayoutCatalog
            #
            if cfg.layouts:
                log( u'Layout Catalog "%s"' % cur_fmpxml )
                for layout_catalog in basenode.getiterator ( "LayoutCatalog" ):
                    groups = []
                    get_layouts_and_groups(cfg,
                                           cur_fmpxml,
                                           cur_db,
                                           cur_fmpbasename,
                                           layout_catalog,
                                           groups,
                                           exportfolder,
                                           1)
                # collect references to fields, CFs, value lists, merge fields,
                # scripts, TOs, FileReferences

            #
            # account catalog
            #
            if cfg.accounts:
                log( u'Accounts for "%s"' % cur_fmpxml )
                for acc_cat in basenode.getiterator ( "AccountCatalog" ):
                    for acc in acc_cat.getchildren():
                        acc_attrib = acc.attrib
                        s = ElementTree.tostring(acc, encoding="utf-8", method="xml")

                        objectID = acc_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "Accounts",
                                               acc_attrib.get("name", "NONAME"),
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists, TOs, FileReferences

            #
            # script catalog
            #
            if cfg.scripts:
                log( u'Scripts for "%s"' % cur_fmpxml )
                for scpt_cat in basenode.getiterator ( "ScriptCatalog" ):
                    groups = []
                    namecache = [{},{}]
                    get_scripts_and_groups(cfg,
                                           cur_fmpxml,
                                           cur_db,
                                           cur_fmpbasename,
                                           scpt_cat,
                                           exportfolder,
                                           groups,
                                           namecache,
                                           1)
                # collect references to fields, CFs, value lists, scripts,
                # TOs, FileReferences

            #
            # custom function catalog
            #
            #
            if cfg.customfunctions:
                log( u'Custom Functions for "%s"' % cur_fmpxml )
                for cf_cat in basenode.getiterator ( "CustomFunctionCatalog" ):
                    groups = []
                    for cf in cf_cat.getchildren():
                        cf_attrib = cf.attrib
                        s = ElementTree.tostring(cf, encoding="utf-8", method="xml")

                        objectID = cf_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "CustomFunctions",
                                               cf_attrib.get("name", "NONAME"),
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists,TOs, FileReferences

            #
            # PrivilegesCatalog
            #
            if cfg.privileges:
                log( u'Privileges for "%s"' % cur_fmpxml )
                for pv_cat in basenode.getiterator( "PrivilegesCatalog" ):
                    for pv in pv_cat.getchildren():
                        pv_attrib = pv.attrib
                        s = ElementTree.tostring(pv, encoding="utf-8", method="xml")

                        objectID = pv_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "Privileges",
                                               pv_attrib.get("name", "NONAME"),
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists, TOs, FileReferences

            #
            # ExtendedPrivilegeCatalog
            #
            if cfg.extendedprivileges:
                log( u'Extended Privileges for "%s"' % cur_fmpxml )
                for epv_cat in basenode.getiterator( "ExtendedPrivilegeCatalog" ):
                    for epv in epv_cat.getchildren():
                        epv_attrib = epv.attrib
                        s = ElementTree.tostring(epv, encoding="utf-8", method="xml")

                        objectID = epv_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "ExtendedPrivileges",
                                               epv_attrib.get("name", "NONAME"),
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists, TOs, FileReferences

            #
            # AuthFileCatalog
            #
            if cfg.authfile:
                log( u'AuthFile Catalog "%s"' % cur_fmpxml )
                for authfile_catalog in basenode.getiterator ( "AuthFileCatalog" ):
                    groups = []
                    get_authfilecatalog(cfg,
                                           cur_fmpxml,
                                           cur_db,
                                           cur_fmpbasename,
                                           authfile_catalog,
                                           groups,
                                           exportfolder,
                                           1)

            #
            # ExternalDataSourcesCatalog
            #
            if cfg.externaldatasources:
                log( u'ExternalDataSources Catalog "%s"' % cur_fmpxml )
                for externaldatasource in basenode.getiterator ( "ExternalDataSourcesCatalog" ):
                    groups = []
                    get_externaldatasources(cfg,
                                           cur_fmpxml,
                                           cur_db,
                                           cur_fmpbasename,
                                           externaldatasource,
                                           groups,
                                           exportfolder,
                                           1)

            #
            # ThemeCatalog
            #
            if cfg.themecatalog:
                log( u'Theme Catalog "%s"' % cur_fmpxml )
                for theme in basenode.getiterator ( "ThemeCatalog" ):
                    groups = []
                    get_themecatalog(cfg,
                                       cur_fmpxml,
                                       cur_db,
                                       cur_fmpbasename,
                                       theme,
                                       groups,
                                       exportfolder,
                                       1)

            #
            # BaseDirectoryCatalog
            #
            if cfg.basedirectory:
                log( u'BaseDirectory Catalog "%s"' % cur_fmpxml )
                for basedir in basenode.getiterator ( "BaseDirectoryList" ):
                    groups = []
                    get_basedirectories(cfg,
                                       cur_fmpxml,
                                       cur_db,
                                       cur_fmpbasename,
                                       basedir,
                                       groups,
                                       exportfolder,
                                       1)

            #
            # CustomMenuCatalog
            #
            if cfg.custommenus:
                log( u'Custom Menus for "%s"' % cur_fmpxml )
                for cm_cat in basenode.getiterator( "CustomMenuCatalog" ):
                    for cm in cm_cat.getchildren():
                        cm_attrib = cm.attrib
                        s = ElementTree.tostring(cm, encoding="utf-8", method="xml")

                        objectID = cm_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "CustomMenus",
                                               cm_attrib.get("name", "NONAME"),
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists, TOs, FileReferences

            #
            # CustomMenuSetCatalog
            #
            if cfg.custommenusets:
                log( u'Custom Menu Sets for "%s"' % cur_fmpxml )
                for cms_cat in basenode.getiterator( "CustomMenuSetCatalog" ):
                    for cms in cms_cat.getchildren():
                        cms_attrib = cms.attrib
                        s = ElementTree.tostring(cms, encoding="utf-8", method="xml")

                        objectID = cms_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "CustomMenuSets",
                                               cms_attrib.get("name", "NONAME"),
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists, TOs, FileReferences

            #
            # ValueListCatalog
            #
            if cfg.valueLists:
                log('Value Lists for "%s"' % cur_fmpxml )
                for vl_cat in basenode.getiterator( "ValueListCatalog" ):
                    for vl in vl_cat.getchildren():
                        vl_attrib = vl.attrib
                        s = ElementTree.tostring(vl, encoding="utf-8", method="xml")

                        objectID = vl_attrib.get("id", "0")
                        if cfg.ignoreFilenameIDs:
                            objectID = ""
                        path = xmlexportfolder(exportfolder,
                                               cur_fmpbasename,
                                               "ValueLists",
                                               vl_attrib.get("name", "NONAME"),
                                               objectID)
                        f = open(path, "wb")
                        f.write( s )
                        f.close()
                # collect references to fields, CFs, value lists, TOs, FileReferences

            if gCancel:
                time.sleep(0.3)
                log("\n\n####  CANCELLED.  ####")
                return
            else:
                print()
                print()

        #
        # References
        #

        # objects

        path = xmlexportfolder(exportfolder,
                               "",
                               "References",
                               "id_object",
                               ext=".tab")
        f = io.open(path, "wb")
        s = u"%i\t%s\n"
        s = u"%i\t%s\t%s\t%s\t%s\t%s\n"
        keys = list( gREF.objectsReverse.keys() )

        # pdb.set_trace()

        keys.sort()
        for key in keys:
            v = gREF.objectsReverse[ key ]
            s4 = s5 = u""
            s1, s2, s3 = v[:3]
            if len(v) == 4:
                s4 = v[3]
            elif len(v) == 5:
                s4 = v[3]
                s5 = v[4]

            t = s % (key, s1, s2, s3, s4, s5)
            try:
                f.write( t.encode("utf-8") )
            except Exception as err:
                print()
                print( err )
                pdb.set_trace()
                print()

        f.close()

        # pdb.set_trace()

        # filemakerAttributes
        path = xmlexportfolder(exportfolder,
                               "",
                               "References",
                               "objid_name_fmpattribute",
                               ext=".tab")
        f = io.open(path, "wb")
        s = u"%s\t%s\t%s\n"
        for objid in gREF.filemakerAttributes:
            obj = gREF.filemakerAttributes[objid]
            for name in obj:
                v = obj[name]
                t = s % (objid, name, v)
                try:
                    f.write( t.encode("utf-8") )
                except Exception as err:
                    print()
                    print( err )
                    pdb.set_trace()
                    print()

        f.close()

        # pdb.set_trace()

        # references
        path = xmlexportfolder(exportfolder,
                               "",
                               "References",
                               "objid_objid_reference",
                               ext=".tab")
        f = io.open(path, "wb")
        s = u"%i\t%i\n"
        keys = gREF.references.keys()
        #keys.sort()
        for r1 in keys:
            referrers = gREF.references[r1]
            #referrers.sort()
            for r2 in referrers:
                t = s % (r1, r2)
                try:
                    f.write( t.encode("utf-8") )
                except Exception as err:
                    print()
                    print( err )
                    print()
                    pdb.set_trace()

        f.close()

        # pdb.set_trace()

        time.sleep(0.3)
        stoptime = time.time()

        t = "\nRuntime %.4f\n\n####  FINISHED.  ####\n\n"
        log(t % ( round(stoptime - starttime, 4), ))

        # pdb.set_trace()
        print( "FileReferences" )
        pp( gREF.fileReferences )
        print

    if __name__ == '__main__':

        infiles = sys.argv[1:]
        for f in infiles:
            f = os.path.abspath( os.path.expanduser(f) )
            folder, filename = os.path.split( f )

            cfg = Config.Config()

            # if run from terminal, customize here
            cfg.accounts = True
            cfg.assets = True
            cfg.basetables = True
            cfg.customfunctions = True
            cfg.custommenus = True
            cfg.custommenusets = True
            cfg.privileges = True
            cfg.extendedprivileges = True
            cfg.filereferences = True
            cfg.layouts = True
            cfg.layoutGroups = False
            cfg.layoutOrder = False
            cfg.relationships = True
            cfg.scripts = True
            cfg.valueLists = True

            cfg.scriptGroups = False
            cfg.scriptOrder = False

            cfg.ignoreFilenameIDs = False

            # do not customize these
            cfg.summaryfile = f
            cfg.exportfolder = os.path.join( folder, "Exports")
            cfg.logfunction = logfunction

            main( cfg )
```