'''

CloudService XBMC Plugin

Copyright (C) 2013-2014 ddurdle

This program is free software: you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License

along with this program. If not, see <http://www.gnu.org/licenses/>.

'''

# cloudservice - required python modules

import sys

import urllib

import re

import os

# cloudservice - standard XBMC modules

import xbmc, xbmcgui, xbmcplugin, xbmcaddon, xbmcvfs

# common routines

from resources.lib import kodi\_common

# global variables

import addon\_parameters

addon = addon\_parameters.addon

cloudservice2 = addon\_parameters.cloudservice2

cloudservice1 = addon\_parameters.cloudservice1

#\*\*\* testing - gdrive

from resources.lib import tvWindow

from resources.lib import gSpreadsheets

from resources.lib import gSheets\_api4

##\*\*

# cloudservice - standard modules

#from resources.lib import gdrive

#from resources.lib import gdrive\_api2

from resources.lib import cloudservice

from resources.lib import authorization

from resources.lib import folder

from resources.lib import file

from resources.lib import offlinefile

from resources.lib import package

from resources.lib import mediaurl

from resources.lib import crashreport

from resources.lib import gPlayer

from resources.lib import settings

from resources.lib import cache

from resources.lib import TMDB

#global variables

PLUGIN\_URL = sys.argv[0]

plugin\_handle = int(sys.argv[1])

plugin\_queries = settings.parse\_query(sys.argv[2][1:])

addon\_dir = xbmc.translatePath( addon.getAddonInfo('path') )

kodi\_common.debugger()

# cloudservice - create settings module

settings = settings.settings(addon)

# retrieve settings

user\_agent = settings.getSetting('user\_agent')

#obsolete, replace, revents audio from streaming

#if user\_agent == 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)':

# addon.setSetting('user\_agent', 'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.38 Safari/532.0')

mode = settings.getParameter('mode','main')

# make mode case-insensitive

mode = mode.lower()

#\*\*\* old - gdrive

# allow for playback of public videos without authentication

if (mode == 'streamurl'):

authenticate = False

else:

authenticate = True

##\*\*

instanceName = ''

try:

instanceName = (plugin\_queries['instance']).lower()

except:

pass

# cloudservice - content type

contextType = settings.getParameter('content\_type')

#support encfs?

encfs = settings.getParameter('encfs', False)

contentType = kodi\_common.getContentType(contextType,encfs)

xbmcplugin.addSortMethod(int(sys.argv[1]), xbmcplugin.SORT\_METHOD\_LABEL)

# xbmcplugin.addSortMethod(int(sys.argv[1]), xbmcplugin.SORT\_METHOD\_TRACKNUM)

xbmcplugin.addSortMethod(int(sys.argv[1]), xbmcplugin.SORT\_METHOD\_SIZE)

numberOfAccounts = kodi\_common.numberOfAccounts(addon\_parameters.PLUGIN\_NAME)

invokedUsername = settings.getParameter('username')

# cloudservice - utilities

###

if mode == 'dummy' or mode == 'delete' or mode == 'enroll':

kodi\_common.accountActions(addon, addon\_parameters.PLUGIN\_NAME, mode, instanceName, numberOfAccounts)

#create strm files

elif mode == 'buildstrm':

silent = settings.getParameter('silent', settings.getSetting('strm\_silent',0))

if silent == '':

silent = 0

try:

path = settings.getSetting('strm\_path')

except:

path = xbmcgui.Dialog().browse(0,addon.getLocalizedString(30026), 'files','',False,False,'')

addon.setSetting('strm\_path', path)

if path == '':

path = xbmcgui.Dialog().browse(0,addon.getLocalizedString(30026), 'files','',False,False,'')

addon.setSetting('strm\_path', path)

if path != '':

returnPrompt = xbmcgui.Dialog().yesno(addon.getLocalizedString(30000), addon.getLocalizedString(30027) + '\n'+path + '?')

if path != '' and returnPrompt:

if silent != 2:

try:

pDialog = xbmcgui.DialogProgressBG()

pDialog.create(addon.getLocalizedString(30000), 'Building STRMs...')

except:

pass

url = settings.getParameter('streamurl')

url = re.sub('---', '&', url)

title = settings.getParameter('title')

type = int(settings.getParameter('type', 0))

if url != '':

filename = path + '/' + title+'.strm'

strmFile = xbmcvfs.File(filename, "w")

strmFile.write(url+'\n')

strmFile.close()

else:

folderID = settings.getParameter('folder')

filename = settings.getParameter('filename')

title = settings.getParameter('title')

invokedUsername = settings.getParameter('username')

encfs = settings.getParameter('encfs', False)

encryptedPath = settings.getParameter('epath', '')

dencryptedPath = settings.getParameter('dpath', '')

if folderID != '':

count = 1

loop = True

while loop:

instanceName = addon\_parameters.PLUGIN\_NAME+str(count)

try:

username = settings.getSetting(instanceName+'\_username')

if username == invokedUsername:

#let's log in

if ( settings.getSettingInt(instanceName+'\_type',0)==0):

service = cloudservice1(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

else:

service = cloudservice2(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

loop = False

except:

service = cloudservice1(PLUGIN\_URL,addon,instanceName, user\_agent)

break

if count == numberOfAccounts:

try:

service

except NameError:

#fallback on first defined account

if ( settings.getSettingInt(instanceName+'\_type',0)==0):

service = cloudservice1(PLUGIN\_URL,addon,addon\_parameters.PLUGIN\_NAME+'1', user\_agent, settings)

else:

service = cloudservice2(PLUGIN\_URL,addon,addon\_parameters.PLUGIN\_NAME+'1', user\_agent, settings)

break

count = count + 1

# encfs -- extract filename

if encfs:

extrapulatedFolderName = re.compile('([^/]+)/$')

titleDecrypted = extrapulatedFolderName.match(dencryptedPath)

if titleDecrypted is not None:

title = titleDecrypted.group(1)

if addon\_parameters.spreadsheet and service.cloudResume == '2':

spreadsheetFile = xbmcvfs.File(path + '/spreadsheet.tab', "w")

service.buildSTRM(path + '/'+title,folderID, contentType=contentType, pDialog=pDialog, epath=encryptedPath, dpath=dencryptedPath, encfs=encfs, spreadsheetFile=spreadsheetFile)

spreadsheetFile.close()

else:

service.buildSTRM(path + '/'+title,folderID, contentType=contentType, pDialog=pDialog, epath=encryptedPath, dpath=dencryptedPath, encfs=encfs)

elif filename != '':

if encfs:

values = {'title': title, 'encfs': 'True', 'epath': encryptedPath, 'dpath': dencryptedPath, 'filename': filename, 'username': invokedUsername}

# encfs -- extract filename

extrapulatedFileName = re.compile('.\*?/([^/]+)$')

titleDecrypted = extrapulatedFileName.match(dencryptedPath)

if titleDecrypted is not None:

title = titleDecrypted.group(1)

else:

values = {'title': title, 'filename': filename, 'username': invokedUsername}

if type == 1:

url = PLUGIN\_URL+'?mode=audio&'+urllib.urlencode(values)

else:

url = PLUGIN\_URL+'?mode=video&'+urllib.urlencode(values)

filename = path + '/' + title+'.strm'

strmFile = xbmcvfs.File(filename, "w")

strmFile.write(url+'\n')

strmFile.close()

else:

count = 1

while True:

instanceName = addon\_parameters.PLUGIN\_NAME+str(count)

username = settings.getSetting(instanceName+'\_username')

if username != '' and username == invokedUsername:

if ( settings.getSettingInt(instanceName+'\_type',0)==0):

service = cloudservice1(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

else:

service = cloudservice2(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

service.buildSTRM(path + '/'+username, contentType=contentType, pDialog=pDialog, epath=encryptedPath, dpath=dencryptedPath, encfs=encfs)

if count == numberOfAccounts:

#fallback on first defined account

try:

service

except NameError:

#fallback on first defined account

if ( settings.getSettingInt(instanceName+'\_type',0)==0):

service = cloudservice1(PLUGIN\_URL,addon,addon\_parameters.PLUGIN\_NAME+'1', user\_agent, settings)

else:

service = cloudservice2(PLUGIN\_URL,addon,addon\_parameters.PLUGIN\_NAME+'1', user\_agent, settings)

break

count = count + 1

if silent != 2:

try:

pDialog.update(100)

pDialog.close()

except:

pass

if silent == 0:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30028))

xbmcplugin.endOfDirectory(plugin\_handle)

###

###

#STRM playback without instance name; use default

if invokedUsername == '' and instanceName == '' and (mode == 'video' or mode == 'audio'):

instanceName = addon\_parameters.PLUGIN\_NAME + str(settings.getSetting('account\_default', 1))

instanceName = kodi\_common.getInstanceName(addon, addon\_parameters.PLUGIN\_NAME, mode, instanceName, invokedUsername, numberOfAccounts, contextType)

service = None

if instanceName is None and (mode == 'index' or mode == 'main' or mode == 'offline'):

service = None

elif instanceName is None:

service = cloudservice2(PLUGIN\_URL,addon,'', user\_agent, settings, authenticate=False)

elif settings.getSettingInt(instanceName+'\_type',0)==0 :

service = cloudservice1(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

else:

service = cloudservice2(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

#create strm files

if mode == 'buildstrm2':

import time

currentDate = time.strftime("%Y%m%d")

try:

path = settings.getSetting('strm\_path')

except:

pass

if path != '':

try:

pDialog = xbmcgui.DialogProgressBG()

pDialog.create(addon.getLocalizedString(30000), 'Building STRMs...')

except:

pass

#service = gdrive\_api2.gdrive(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

# try:

addon.setSetting(instanceName + '\_changedate', currentDate)

service.buildSTRM2(path, contentType=contentType, pDialog=pDialog)

# except:

# pass

try:

pDialog.update(100)

pDialog.close()

except:

pass

xbmcplugin.endOfDirectory(plugin\_handle)

# options menu

#if mode == 'main':

# addMenu(PLUGIN\_URL+'?mode=options','<< '+addon.getLocalizedString(30043)+' >>')

if mode == 'offline':

title = settings.getParameter('title')

folderID = settings.getParameter('folder')

folderName = settings.getParameter('foldername')

mediaItems = kodi\_common.getOfflineFileList(settings.getSetting('cache\_folder'))

if mediaItems:

for offlinefile in mediaItems:

kodi\_common.addOfflineMediaFile(offlinefile)

elif service is None:

xbmcplugin.endOfDirectory(plugin\_handle)

#cloud\_db actions

elif mode == 'cloud\_db':

title = settings.getParameter('title')

folderID = settings.getParameter('folder')

folderName = settings.getParameter('foldername')

filename = settings.getParameter('filename')

action = settings.getParameter('action')

mediaFile = file.file(filename, title, '', 0, '','')

mediaFolder = folder.folder(folderID,folderName)

package=package.package(mediaFile,mediaFolder)

# TESTING

if addon\_parameters.spreadsheet and service.cloudResume == '2':

if service.worksheetID == '':

try:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

spreadsheets = service.gSpreadsheet.getSpreadsheetList()

except:

pass

for title in spreadsheets.iterkeys():

if title == 'CLOUD\_DB':

worksheets = service.gSpreadsheet.getSpreadsheetWorksheets(spreadsheets[title])

for worksheet in worksheets.iterkeys():

if worksheet == 'db':

service.worksheetID = worksheets[worksheet]

addon.setSetting(instanceName + '\_spreadsheet', service.worksheetID)

break

break

# TESTING

if addon\_parameters.spreadsheet and service.cloudResume == '2':

if service.gSpreadsheet is None:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

if action == 'watch':

service.gSpreadsheet.setMediaStatus(service.worksheetID,package, watched=1)

xbmc.executebuiltin("XBMC.Container.Refresh")

elif action == 'queue':

package.folder.id = 'QUEUED'

service.gSpreadsheet.setMediaStatus(service.worksheetID,package)

elif action == 'recentwatched' or action == 'recentstarted' or action == 'library' or action == 'queued':

mediaItems = service.gSpreadsheet.updateMediaPackage(service.worksheetID, criteria=action)

#ensure that folder view playback

if contextType == '':

contextType = 'video'

if mediaItems:

for item in mediaItems:

if item.file is None:

service.addDirectory(item.folder, contextType=contextType)

else:

service.addMediaFile(item, contextType=contextType)

service.updateAuthorization(addon)

#cloud\_db actions

elif mode == 'cloud\_dbtest':

title = settings.getParameter('title')

folderID = settings.getParameter('folder')

folderName = settings.getParameter('foldername')

filename = settings.getParameter('filename')

action = settings.getParameter('action')

# s = gSheets\_api4.gSheets\_api4(service,addon, user\_agent)

# s.createSpreadsheet()

# s.addRows()

if action == 'library\_menu':

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_genre&content\_type='+str(contextType),'Genre')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_year&content\_type='+str(contextType),'Year')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_title&content\_type='+str(contextType),'Title')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_country&content\_type='+str(contextType),'Countries')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_director&content\_type='+str(contextType),'Directors')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_studio&content\_type='+str(contextType),'Studio')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_resolution&content\_type='+str(contextType),'Quality (Resolution)')

else:

mediaFile = file.file(filename, title, '', 0, '','')

mediaFolder = folder.folder(folderID,folderName)

package=package.package(mediaFile,mediaFolder)

spreadsheet = None

# TESTING

if addon\_parameters.spreadsheet:

try:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

spreadsheets = service.gSpreadsheet.getSpreadsheetList()

except:

pass

for t in spreadsheets.iterkeys():

if t == 'Movie2':

worksheets = service.gSpreadsheet.getSpreadsheetWorksheets(spreadsheets[t])

for worksheet in worksheets.iterkeys():

if worksheet == 'db':

spreadsheet = worksheets[worksheet]

break

break

# TESTING

if addon\_parameters.spreadsheet:

if service.gSpreadsheet is None:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

if action == 'watch':

service.gSpreadsheet.setMediaStatus(service.worksheetID,package, watched=1)

xbmc.executebuiltin("XBMC.Container.Refresh")

elif action == 'queue':

package.folder.id = 'QUEUED'

service.gSpreadsheet.setMediaStatus(service.worksheetID,package)

elif action == 'genre' or action == 'year' or action == 'title' or action == 'country' or action == 'director' or action == 'studio' or action == 'recentstarted' or 'library' in action or action == 'queued':

if action == 'genre':

mediaItems = service.gSpreadsheet.getMovies(spreadsheet, genre=title)

elif action == 'year':

mediaItems = service.gSpreadsheet.getMovies(spreadsheet, year=title)

elif action == 'title':

mediaItems = service.gSpreadsheet.getMovies(spreadsheet, title=title)

elif action == 'resolution':

mediaItems = service.gSpreadsheet.getMovies(spreadsheet, resolution=title)

elif action == 'country':

mediaItems = service.gSpreadsheet.getMovies(spreadsheet, country=title)

elif action == 'director':

mediaItems = service.gSpreadsheet.getMovies(spreadsheet, director=title)

elif action == 'studio':

mediaItems = service.gSpreadsheet.getMovies(spreadsheet, studio=title)

elif action == 'library\_title':

mediaItems = service.gSpreadsheet.getTitle(spreadsheet)

elif action == 'library\_genre':

mediaItems = service.gSpreadsheet.getGenre(spreadsheet)

elif action == 'library\_year':

mediaItems = service.gSpreadsheet.getYear(spreadsheet)

elif action == 'library\_country':

mediaItems = service.gSpreadsheet.getCountries(spreadsheet)

elif action == 'library\_director':

mediaItems = service.gSpreadsheet.getDirector(spreadsheet)

elif action == 'library\_studio':

mediaItems = service.gSpreadsheet.getStudio(spreadsheet)

elif action == 'library\_resolution':

mediaItems = service.gSpreadsheet.getResolution(spreadsheet)

#ensure that folder view playback

if contextType == '':

contextType = 'video'

tmdb= TMDB.TMDB(service,addon, user\_agent)

if mediaItems:

for item in mediaItems:

if item.file is None:

service.addDirectory(item.folder, contextType=contextType)

else:

# movieID = tmdb.movieSearch(item.file.title,item.file.year)

# tmdb.movieDetails(movieID)

service.addMediaFile(item, contextType=contextType)

service.updateAuthorization(addon)

#dump a list of videos available to play

elif mode == 'main' or mode == 'index':

folderID = settings.getParameter('folder', False)

folderName = settings.getParameter('foldername', False)

#ensure that folder view playback

if contextType == '':

contextType = 'video'

# display option for all Videos/Music/Photos, across gdrive

#\*\* gdrive specific

if mode == 'main':

if ('gdrive' in addon\_parameters.PLUGIN\_NAME):

if contentType in (2,4,7):

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=ALL&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+' '+addon.getLocalizedString(30030)+']')

elif contentType == 1:

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=VIDEOMUSIC&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+' '+addon.getLocalizedString(30031)+']')

elif contentType == 0:

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=VIDEO&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+' '+addon.getLocalizedString(30025)+']')

elif contentType == 3:

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=MUSIC&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+' '+addon.getLocalizedString(30094)+']')

elif contentType == 5:

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=PHOTO&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+' '+addon.getLocalizedString(30034)+']')

elif contentType == 6:

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=PHOTOMUSIC&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+' '+addon.getLocalizedString(30032)+']')

folderID = 'root'

if ('gdrive' in addon\_parameters.PLUGIN\_NAME):

# if (service.protocol != 2):

# kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=STARRED-FILES&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+ ' '+addon.getLocalizedString(30095)+']')

# kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=STARRED-FOLDERS&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+ ' '+addon.getLocalizedString(30096)+']')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=SHARED&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+ ' '+addon.getLocalizedString(30098)+']')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=index&folder=STARRED-FILESFOLDERS&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30018)+ ' '+addon.getLocalizedString(30097)+']')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=search&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30111)+']')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=buildstrm2&instance='+str(service.instanceName)+'&content\_type='+str(contextType),'<Testing - manual run of change tracking build STRM>')

if addon\_parameters.testing\_features:

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_dbtest&instance='+str(service.instanceName)+'&action=library\_menu&content\_type='+str(contextType),'[MOVIES]')

#CLOUD\_DB

if 'gdrive' in addon\_parameters.PLUGIN\_NAME and service.gSpreadsheet is not None:

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_db&action=recentstarted&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30177)+' recently started]')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_db&action=recentwatched&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30177)+' recently watched]')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_db&action=library&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30177)+' library]')

kodi\_common.addMenu(PLUGIN\_URL+'?mode=cloud\_db&action=queued&instance='+str(service.instanceName)+'&content\_type='+contextType,'['+addon.getLocalizedString(30177)+' queued]')

##\*\*

# cloudservice - validate service

try:

service

except NameError:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30051), addon.getLocalizedString(30052))

xbmc.log(addon.getLocalizedString(30050)+ addon\_parameters.PLUGIN\_NAME+'-login', xbmc.LOGERROR)

xbmcplugin.endOfDirectory(plugin\_handle)

#if encrypted, get everything(as encrypted files will be of type application/ostream)

if encfs:

settings.setEncfsParameters()

encryptedPath = settings.getParameter('epath', '')

dencryptedPath = settings.getParameter('dpath', '')

encfs\_source = settings.encfsSource

encfs\_target = settings.encfsTarget

encfs\_inode = settings.encfsInode

mediaItems = service.getMediaList(folderID,contentType=8)

if mediaItems:

dirListINodes = {}

fileListINodes = {}

#create the files and folders for decrypting file/folder names

for item in mediaItems:

if item.file is None:

xbmcvfs.mkdir(encfs\_source + str(encryptedPath))

xbmcvfs.mkdir(encfs\_source + str(encryptedPath) + str(item.folder.title) + '/' )

if encfs\_inode == 0:

dirListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath) + str(item.folder.title)).st\_ino()))] = item.folder

else:

dirListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath) + str(item.folder.title)).st\_ctime()))] = item.folder

#service.addDirectory(item.folder, contextType=contextType, encfs=True)

else:

xbmcvfs.mkdir(encfs\_source + str(encryptedPath))

xbmcvfs.mkdir(encfs\_source + str(encryptedPath) + str(item.file.title))

if encfs\_inode == 0:

fileListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath)+ str(item.file.title)).st\_ino()))] = item

else:

fileListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath) + str(item.file.title)).st\_ctime()))] = item

#service.addMediaFile(item, contextType=contextType)

if encfs\_inode > 0:

xbmc.sleep(1000)

if contentType == 9:

mediaList = ['.mp4', '.flv', '.mov', '.webm', '.avi', '.ogg', '.mkv']

elif contentType == 10:

mediaList = ['.mp3', '.flac']

else:# contentType == 11:

mediaList = ['.jpg', '.png']

media\_re = re.compile("|".join(mediaList), re.I)

#examine the decrypted file/folder names for files for playback and dirs for navigation

dirs, files = xbmcvfs.listdir(encfs\_target + str(dencryptedPath) )

for dir in dirs:

index = ''

if encfs\_inode == 0:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPath) + dir).st\_ino())

else:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPath) + dir).st\_ctime())

#we found a directory

if index in dirListINodes.keys():

xbmcvfs.rmdir(encfs\_target + str(dencryptedPath) + dir)

# dirTitle = dir + ' [' +dirListINodes[index].title+ ']'

encryptedDir = dirListINodes[index].title

dirListINodes[index].displaytitle = dir + ' [' +dirListINodes[index].title+ ']'

service.addDirectory(dirListINodes[index], contextType=contextType, encfs=True, dpath=str(dencryptedPath) + str(dir) + '/', epath=str(encryptedPath) + str(encryptedDir) + '/' )

#we found a file

elif index in fileListINodes.keys():

xbmcvfs.rmdir(encfs\_target + str(dencryptedPath) + dir)

fileListINodes[index].file.decryptedTitle = dir

if contentType < 9 or media\_re.search(str(dir)):

service.addMediaFile(fileListINodes[index], contextType=contextType, encfs=True, dpath=str(dencryptedPath) + str(dir), epath=str(encryptedPath) )

# file is already downloaded

for file in files:

index = ''

if encfs\_inode == 0:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPath) + file).st\_ino())

else:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPath) + file).st\_ctime())

if index in fileListINodes.keys():

fileListINodes[index].file.decryptedTitle = file

if contentType < 9 or media\_re.search(str(file)):

service.addMediaFile(fileListINodes[index], contextType=contextType, encfs=True, dpath=str(dencryptedPath) + str(file), epath=str(encryptedPath) )

#xbmc.executebuiltin("XBMC.Container.Refresh")

else:

path = settings.getParameter('epath', '')

# real folder

if folderID != '':

mediaItems = service.getMediaList(folderID,contentType=contentType)

if addon\_parameters.spreadsheet and service.cloudResume == '2':

if service.gSpreadsheet is None:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

if service.worksheetID != '':

service.gSpreadsheet.updateMediaPackageList(service.worksheetID, folderID, mediaItems)

if mediaItems:

for item in sorted(mediaItems):

if item.file is None:

service.addDirectory(item.folder, contextType=contextType, epath=str(path)+ '/' + str(item.folder.title) + '/')

else:

service.addMediaFile(item, contextType=contextType)

# virtual folder; exists in spreadsheet only

# not in use

#elif folderName != '':

service.updateAuthorization(addon)

# NOT IN USE

#\*\* testing - gdrive

elif mode == 'kiosk':

spreadshetModule = settings.getSetting('library', False)

if spreadshetModule:

gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

service.gSpreadsheet = gSpreadsheet

spreadsheets = service.getSpreadsheetList()

channels = []

for title in spreadsheets.iterkeys():

if title == 'TVShows':

worksheets = gSpreadsheet.getSpreadsheetWorksheets(spreadsheets[title])

if 0:

import time

hour = time.strftime("%H")

minute = time.strftime("%M")

weekDay = time.strftime("%w")

month = time.strftime("%m")

day = time.strftime("%d")

for worksheet in worksheets.iterkeys():

if worksheet == 'schedule':

channels = gSpreadsheet.getChannels(worksheets[worksheet])

ret = xbmcgui.Dialog().select(addon.getLocalizedString(30112), channels)

shows = gSpreadsheet.getShows(worksheets[worksheet] ,channels[ret])

showList = []

for show in shows:

showList.append(shows[show][6])

ret = xbmcgui.Dialog().select(addon.getLocalizedString(30112), showList)

for worksheet in worksheets.iterkeys():

if worksheet == 'data':

episodes = gSpreadsheet.getVideo(worksheets[worksheet] ,showList[ret])

#player = gPlayer.gPlayer()

#player.setService(service)

player.setContent(episodes)

player.setWorksheet(worksheets['data'])

player.next()

while not player.isExit:

xbmc.sleep(5000)

else:

for worksheet in worksheets.iterkeys():

if worksheet == 'db':

episodes = gSpreadsheet.getMedia(worksheets[worksheet], service.getRootID())

#player = gPlayer.gPlayer()

#player.setService(service)

# player.setContent(episodes)

player.setWorksheet(worksheets['db'])

player.PlayStream('plugin://plugin.video.'+addon\_parameters.PLUGIN\_NAME+'-testing/?mode=video&instance='+str(service.instanceName)+'&title='+episodes[0][3], None,episodes[0][7],episodes[0][2])

#player.next()

while not player.isExit:

player.saveTime()

xbmc.sleep(5000)

##\*\* not in use

elif mode == 'photo':

title = settings.getParameter('title',0)

title = re.sub('/', '\_', title) #remap / from titles (google photos)

docid = settings.getParameter('filename')

folder = settings.getParameter('folder',0)

encfs = settings.getParameter('encfs', False)

if encfs:

settings.setEncfsParameters()

encryptedPath = settings.getParameter('epath', '')

dencryptedPath = settings.getParameter('dpath', '')

encfs\_source = settings.encfsSource

encfs\_target = settings.encfsTarget

encfs\_inode = settings.encfsInode

# don't redownload if present already

if (not xbmcvfs.exists(str(encfs\_source) + str(encryptedPath) +str(title))):

url = service.getDownloadURL(docid)

service.downloadGeneralFile(url, str(encfs\_source) + str(encryptedPath) +str(title))

xbmc.executebuiltin("XBMC.ShowPicture(\""+str(encfs\_target) + str(dencryptedPath)+"\")")

#item = xbmcgui.ListItem(path=str(encfs\_target) + str(dencryptedPath))

#xbmcplugin.setResolvedUrl(int(sys.argv[1]), True, item)

else:

path = settings.getSetting('photo\_folder')

#workaround for this issue: https://github.com/xbmc/xbmc/pull/8531

if not xbmcvfs.exists(path) and not os.path.exists(path):

path = ''

while path == '':

path = xbmcgui.Dialog().browse(0,addon.getLocalizedString(30038), 'files','',False,False,'')

#workaround for this issue: https://github.com/xbmc/xbmc/pull/8531

if not xbmcvfs.exists(path) and not os.path.exists(path):

path = ''

else:

addon.setSetting('photo\_folder', path)

if (not xbmcvfs.exists(str(path) + '/'+str(folder) + '/')):

xbmcvfs.mkdir(str(path) + '/'+str(folder))

# try:

# xbmcvfs.rmdir(str(path) + '/'+str(folder)+'/'+str(title))

# except:

# pass

# don't redownload if present already

if (not xbmcvfs.exists(str(path) + '/'+str(folder)+'/'+str(title))):

url = service.getDownloadURL(docid)

service.downloadPicture(url, str(path) + '/'+str(folder) + '/'+str(title))

#xbmc.executebuiltin("XBMC.ShowPicture("+str(path) + '/'+str(folder) + '/'+str(title)+")")

#item = xbmcgui.ListItem(path=str(path) + '/'+str(folder) + '/'+str(title))

url = service.getDownloadURL(docid)

item = xbmcgui.ListItem(path=url + '|' + service.getHeadersEncoded())

xbmcplugin.setResolvedUrl(int(sys.argv[1]), True, item)

elif mode == 'downloadfolder':

title = settings.getParameter('title')

folderID = settings.getParameter('folder')

folderName = settings.getParameter('foldername')

encfs = settings.getParameter('encfs', False)

try:

service

except NameError:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30051), addon.getLocalizedString(30052))

xbmc.log(addon.getLocalizedString(30050)+ addon\_parameters.PLUGIN\_NAME + '-login',xbmc.LOGERROR)

xbmcplugin.endOfDirectory(plugin\_handle)

if encfs:

settings.setEncfsParameters()

encryptedPath = settings.getParameter('epath', '')

dencryptedPath = settings.getParameter('dpath', '')

encfs\_source = settings.encfsSource

encfs\_target = settings.encfsTarget

encfs\_inode = settings.encfsInode

else:

path = settings.getParameter('epath', '/')

if encfs:

mediaItems = service.getMediaList(folderName=folderID, contentType=8)

path = str(encfs\_source) + str(encryptedPath)

else:

mediaItems = service.getMediaList(folderName=folderID, contentType=contentType)

path = str(settings.getSetting('photo\_folder')) + str(path)

if mediaItems:

progress = xbmcgui.DialogProgressBG()

progressBar = len(mediaItems)

progress.create(addon.getLocalizedString(30092), '')

count=0

if not xbmcvfs.exists(path) and not os.path.exists(path):

xbmcvfs.mkdirs(path)

for item in mediaItems:

count = count + 1

if item.file is not None:

progress.update((int)(float(count)/len(mediaItems)\*100),addon.getLocalizedString(30092), str(item.file.title))

service.downloadGeneralFile(item.getMediaURL(),str(path) + str(item.file.title) )

# elif item.folder is not None:

# # create path if doesn't exist

# if (not xbmcvfs.exists(str(path) + '/'+str(folder) + '/')):

# xbmcvfs.mkdir(str(path) + '/'+str(folder))

progress.close()

elif mode == 'slideshow':

folder = settings.getParameter('folder',0)

title = settings.getParameter('title',0)

encfs = settings.getParameter('encfs', False)

if encfs:

settings.setEncfsParameters()

encfs\_source = settings.encfsSource

encfs\_target = settings.encfsTarget

encfs\_inode = settings.encfsInode

if (not xbmcvfs.exists(str(encfs\_target) + '/'+str(folder) + '/')):

xbmcvfs.mkdir(str(encfs\_target) + '/'+str(folder))

folderINode = ''

if encfs\_inode == 0:

folderINode = str(xbmcvfs.Stat(encfs\_target + '/' + str(folder)).st\_ino())

else:

folderINode = str(xbmcvfs.Stat(encfs\_target + '/' + str(folder)).st\_ctime())

mediaItems = service.getMediaList(folderName=folder, contentType=8)

if mediaItems:

dirs, filesx = xbmcvfs.listdir(encfs\_source)

for dir in dirs:

index = ''

if encfs\_inode == 0:

index = str(xbmcvfs.Stat(encfs\_source + '/' + dir).st\_ino())

else:

index = str(xbmcvfs.Stat(encfs\_source + '/' + dir).st\_ctime())

if index == folderINode:

progress = xbmcgui.DialogProgressBG()

progress.create(addon.getLocalizedString(30035), 'Preparing list...')

count=0

for item in mediaItems:

if item.file is not None:

count = count + 1;

progress.update((int)(float(count)/len(mediaItems)\*100),addon.getLocalizedString(30035), item.file.title)

if (not xbmcvfs.exists(str(encfs\_source) + '/'+str(dir)+'/'+str(item.file.title))):

service.downloadGeneralFile(item.mediaurl.url,str(encfs\_source) + '/'+str(dir)+ '/'+str(item.file.title))

if encfs\_inode > 0:

xbmc.sleep(100)

progress.close()

xbmc.executebuiltin("XBMC.SlideShow(\""+str(encfs\_target) + '/'+str(folder)+"/\")")

elif 0:

path = settings.getSetting('photo\_folder')

#workaround for this issue: https://github.com/xbmc/xbmc/pull/8531

if not xbmcvfs.exists(path) and not os.path.exists(path):

path = ''

while path == '':

path = xbmcgui.Dialog().browse(0,addon.getLocalizedString(30038), 'files','',False,False,'')

#workaround for this issue: https://github.com/xbmc/xbmc/pull/8531

if not xbmcvfs.exists(path) and not os.path.exists(path):

path = ''

else:

addon.setSetting('photo\_folder', path)

# create path if doesn't exist

if (not xbmcvfs.exists(str(path) + '/'+str(folder) + '/')):

xbmcvfs.mkdir(str(path) + '/'+str(folder))

mediaItems = service.getMediaList(folderName=folder, contentType=5)

if mediaItems:

progress = xbmcgui.DialogProgressBG()

progress.create(addon.getLocalizedString(30035), 'Preparing list...')

count=0

for item in mediaItems:

if item.file is not None:

count = count + 1;

progress.update((int)(float(count)/len(mediaItems)\*100),addon.getLocalizedString(30035), item.file.title)

service.downloadGeneralFile(item.mediaurl.url,str(path) + '/'+str(folder)+ '/'+item.file.title)

#xbmc.executebuiltin("XBMC.SlideShow("+str(path) + '/'+str(folder)+"/)")

progress.close()

xbmc.executebuiltin("XBMC.SlideShow(\""+str(path) + '/'+str(folder)+"/\")")

#else:

# xbmc.executebuiltin("XBMC.SlideShow("+str(path) + '/'+str(folder)+"/)")

###

# for video files

# force stream - play a video given its url

###

elif mode == 'streamurl':

url = settings.getParameter('url',0)

title = settings.getParameter('title')

promptQuality = settings.getSetting('prompt\_quality', True)

mediaURLs = service.getPublicStream(url)

options = []

if mediaURLs:

mediaURLs = sorted(mediaURLs)

for mediaURL in mediaURLs:

options.append(mediaURL.qualityDesc)

if promptQuality:

ret = xbmcgui.Dialog().select(addon.getLocalizedString(30033), options)

else:

ret = 0

playbackURL = mediaURLs[ret].url

if (playbackURL == ''):

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30020),addon.getLocalizedString(30021))

xbmc.log(addon.getAddonInfo('name') + ': ' + addon.getLocalizedString(20021), xbmc.LOGERROR)

else:

# if invoked in .strm or as a direct-video (don't prompt for quality)

item = xbmcgui.ListItem(path=playbackURL+ '|' + service.getHeadersEncoded())

item.setInfo( type="Video", infoLabels={ "Title": mediaURLs[ret].title , "Plot" : mediaURLs[ret].title } )

xbmcplugin.setResolvedUrl(int(sys.argv[1]), True, item)

else:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30020),addon.getLocalizedString(30021))

xbmc.log(addon.getAddonInfo('name') + ': ' + addon.getLocalizedString(20021), xbmc.LOGERROR)

###

# for video files - playback of video

# force stream - play a video given its url

###

#

# legacy (depreicated) - memorycachevideo [given title]

# legacy (depreicated) - play [given title]

# legacy (depreicated) - playvideo [given title]

# legacy (depreicated) - streamvideo [given title]

elif mode == 'audio' or mode == 'video' or mode == 'search' or mode == 'play' or mode == 'memorycachevideo' or mode == 'playvideo' or mode == 'streamvideo':

title = settings.getParameter('title') #file title

filename = settings.getParameter('filename') #file ID

folderID = settings.getParameter('folder') #folder ID

spreadsheetSTRM = settings.getParameter('spreadsheet')

sheetSTRM = settings.getParameter('sheet')

year = settings.getParameter('year')

if sheetSTRM != None and sheetSTRM != '':

if service.gSpreadsheet is None:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

try:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

spreadsheets = service.gSpreadsheet.getSpreadsheetList()

except:

pass

spreadsheet = None

for t in spreadsheets.iterkeys():

if t == 'Movies':

worksheets = service.gSpreadsheet.getSpreadsheetWorksheets(spreadsheets[t])

for worksheet in worksheets.iterkeys():

if worksheet == 'db':

spreadsheet = worksheets[worksheet]

break

break

if spreadsheet != None:

filename = service.gSpreadsheet.getSTRMplaybackMovie(spreadsheet, title, year)

else:

filename = service.gSpreadsheet.getSTRMplaybackMovie('https://spreadsheets.google.com/feeds/list/'+spreadsheetSTRM+'/'+sheetSTRM+'/private/full', title, year)

if folderID == 'False':

folderID = 'SEARCH'

if mode != 'audio':

settings.setVideoParameters()

seek = 0

if settings.seek:

dialog = xbmcgui.Dialog()

seek = dialog.numeric(2, 'Time to seek to', '00:00')

for r in re.finditer('(\d+)\:(\d+)' ,seek, re.DOTALL):

seekHours, seekMins = r.groups()

seek = int(seekMins) + (int(seekHours)\*60)

try:

service

except NameError:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30051), addon.getLocalizedString(30052))

xbmc.log(addon.getLocalizedString(30050)+ addon\_parameters.PLUGIN\_NAME + '-login', xbmc.LOGERROR)

xbmcplugin.endOfDirectory(plugin\_handle)

#settings.setCacheParameters()

if mode == 'memorycachevideo':

settings.play = True

settings.download = True

elif mode == 'playvideo':

settings.play = False

settings.download = False

settings.playOriginal = True

if settings.cache:

settings.download = False

settings.play = False

encfs = settings.getParameter('encfs', False)

#testing

player = gPlayer.gPlayer()

player.setService(service)

resolvedPlayback = True

startPlayback = False

toExit = False

#package = None

if encfs:

settings.setEncfsParameters()

encryptedPath = settings.getParameter('epath', '')

dencryptedPath = settings.getParameter('dpath', '')

encfs\_source = settings.encfsSource

encfs\_target = settings.encfsTarget

encfs\_inode = settings.encfsInode

mediaFile = file.file(filename, title, '', 0, '','')

mediaFolder = folder.folder(folderID,'')

(mediaURLs,package) = service.getPlaybackCall(package=package.package(mediaFile,mediaFolder), title=title, contentType=8)

#(mediaURLs,package) = service.getPlaybackCall(None,title=title)

mediaURL = mediaURLs[0]

playbackTarget = encfs\_target + dencryptedPath

item = xbmcgui.ListItem(package.file.displayTitle(), iconImage=package.file.thumbnail,

thumbnailImage=package.file.thumbnail, path=playbackTarget)

#item.setInfo( type="Video", infoLabels={ "Title": package.file.title , "Plot" : package.file.title } )

# right-click or integrated player (no opening stream dialog...)

if contextType == '':

# for STRM (force resolve) -- resolve-only

if settings.username != '':

resolvedPlayback = True

startPlayback = False

else:

startPlayback = True

# resolve for an opening stream dialog

else:

resolvedPlayback=True

# download if not already cached

# if (not xbmcvfs.exists(str(encfs\_source) + encryptedPath +str(title))):

url = service.getDownloadURL(filename)

## check for SRT

# use folderID, look for files with srt/sub

mediaItems = service.getMediaList(folderID,contentType=8)

encfsSubTitles = []

if mediaItems:

dirListINodes = {}

fileListINodes = {}

#create the files and folders for decrypting file/folder names

for itemx in mediaItems:

if itemx.file is None:

xbmcvfs.mkdir(encfs\_source + str(encryptedPath))

xbmcvfs.mkdir(encfs\_source + str(encryptedPath) + str(itemx.folder.title) + '/' )

if encfs\_inode == 0:

dirListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath) + str(itemx.folder.title)).st\_ino()))] = itemx.folder

else:

dirListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath) + str(itemx.folder.title)).st\_ctime()))] = itemx.folder

#service.addDirectory(item.folder, contextType=contextType, encfs=True)

else:

xbmcvfs.mkdir(encfs\_source + str(encryptedPath))

xbmcvfs.mkdir(encfs\_source + str(encryptedPath) + str(itemx.file.title))

if encfs\_inode == 0:

fileListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath)+ str(itemx.file.title)).st\_ino()))] = itemx

else:

fileListINodes[(str(xbmcvfs.Stat(encfs\_source + str(encryptedPath) + str(itemx.file.title)).st\_ctime()))] = itemx

#service.addMediaFile(itemx, contextType=contextType)

if encfs\_inode > 0:

xbmc.sleep(1000)

mediaList = ['.sub', '.srt']

media\_re = re.compile("|".join(mediaList), re.I)

# encfs -- extract path

extrapulatedPath = re.compile('(.\*?)/[^/]+$')

dencryptedPathWithoutFilename = extrapulatedPath.match(dencryptedPath)

if dencryptedPathWithoutFilename is None:

dencryptedPathWithoutFilename = ''

else:

dencryptedPathWithoutFilename = dencryptedPathWithoutFilename.group(1) + '/'

#examine the decrypted file/folder names for files for playback and dirs for navigation

dirs, files = xbmcvfs.listdir(encfs\_target + str(dencryptedPathWithoutFilename) )

for dir in dirs:

index = ''

if encfs\_inode == 0:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPathWithoutFilename) + dir).st\_ino())

else:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPathWithoutFilename) + dir).st\_ctime())

#we found a file

if index in fileListINodes.keys():

xbmcvfs.rmdir(encfs\_target + str(dencryptedPathWithoutFilename) + dir)

fileListINodes[index].file.decryptedTitle = dir

if media\_re.search(str(dir)):

#we found a subtitle

service.downloadGeneralFile(fileListINodes[index].mediaurl.url, str(encfs\_source) + str(encryptedPath) +str(fileListINodes[index].file.title))

# str(encfs\_target) + str(dencryptedPathWithoutFilename) + str(fileListINodes[index].file.decryptedTitle)

encfsSubTitles.append(str(encfs\_target) + str(dencryptedPathWithoutFilename) + str(fileListINodes[index].file.decryptedTitle))

# file is already downloaded

for file in files:

index = ''

if encfs\_inode == 0:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPathWithoutFilename) + file).st\_ino())

else:

index = str(xbmcvfs.Stat(encfs\_target + str(dencryptedPathWithoutFilename) + file).st\_ctime())

if index in fileListINodes.keys():

fileListINodes[index].file.decryptedTitle = file

if media\_re.search(str(file)):

#we found a subtitle

# service.addMediaFile(fileListINodes[index], contextType=contextType, encfs=True, dpath=str(dencryptedPath) + str(file), epath=str(encryptedPath) )

# service.downloadGeneralFile(fileListINodes[index], package, playbackURL=playbackTarget, folderName=str(encfs\_source) + encryptedPath + str(fileListINodes[index].file.title))

# service.downloadGeneralFile(fileListINodes[index].mediaurl.url, str(encfs\_source) + str(encryptedPath) +str(title))

encfsSubTitles.append(str(encfs\_target) + str(dencryptedPathWithoutFilename) + str(fileListINodes[index].file.decryptedTitle))

if settings.encfsStream or settings.encfsCacheSingle:

## calculate the decrypted name of the file cache.mp4

#creating a cache.mp4 file

fileListINodes = {}

#workaround for this issue: https://github.com/xbmc/xbmc/pull/8531

if not xbmcvfs.exists(encfs\_target + 'encfs.mp4') and not os.path.exists(encfs\_target + 'encfs.mp4'):

xbmcvfs.mkdir(encfs\_target + 'encfs.mp4')

if encfs\_inode == 0:

fileListINodes[(str(xbmcvfs.Stat(encfs\_target + 'encfs.mp4').st\_ino()))] = item

else:

fileListINodes[(str(xbmcvfs.Stat(encfs\_target + 'encfs.mp4').st\_ctime()))] = item

if encfs\_inode > 0:

xbmc.sleep(1000)

dirs, files = xbmcvfs.listdir(encfs\_source)

for dir in dirs:

index = ''

if encfs\_inode == 0:

index = str(xbmcvfs.Stat(encfs\_source + str(dir)).st\_ino())

else:

index = str(xbmcvfs.Stat(encfs\_source + str(dir)).st\_ctime())

#we found a file

if index in fileListINodes.keys():

xbmcvfs.rmdir(encfs\_source + str(dir))

addon.setSetting('encfs\_last', str(encryptedPath) +str(title))

if settings.encfsExp:

service.downloadEncfsFile2(mediaURL, package, playbackURL=encfs\_target + 'encfs.mp4', folderName=str(encfs\_source) + str(dir), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

else:

service.downloadEncfsFile(mediaURL, package, playbackURL=encfs\_target + 'encfs.mp4', folderName=str(encfs\_source) + str(dir), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

#already downloaded (partial or full)

for file in files:

index = ''

if encfs\_inode == 0:

index = str(xbmcvfs.Stat(encfs\_source + str(file)).st\_ino())

else:

index = str(xbmcvfs.Stat(encfs\_source + str(file)).st\_ctime())

#we found a file

if index in fileListINodes.keys():

#resume

if settings.encfsLast == str(encryptedPath) +str(title):

if settings.encfsExp:

service.downloadEncfsFile2(mediaURL, package, playbackURL=encfs\_target + 'encfs.mp4', force=False,folderName=str(encfs\_source) + str(file), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

else:

service.downloadEncfsFile(mediaURL, package, playbackURL=encfs\_target + 'encfs.mp4', force=False,folderName=str(encfs\_source) + str(file), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

#new file

else:

addon.setSetting('encfs\_last', str(encryptedPath) +str(title))

if settings.encfsExp:

service.downloadEncfsFile2(mediaURL, package, playbackURL=encfs\_target + 'encfs.mp4', force=True, folderName=str(encfs\_source) + str(file), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

else:

service.downloadEncfsFile(mediaURL, package, playbackURL=encfs\_target + 'encfs.mp4', force=True, folderName=str(encfs\_source) + str(file), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

else:

#service.downloadEncfsFile2(mediaURL, package, playbackURL=playbackTarget, folderName=str(encfs\_source) + encryptedPath +str(title), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

service.downloadEncfsFile(mediaURL, package, playbackURL=playbackTarget, folderName=str(encfs\_source) + encryptedPath +str(title), playback=resolvedPlayback,item=item, player=player, srt=encfsSubTitles)

#should already be playing by this point, so don't restart it

startPlayback = False

#exists; resolve for an opening stream dialog

# elif resolvedPlayback:

# xbmcplugin.setResolvedUrl(int(sys.argv[1]), True, item)

# need to seek?

#if seek > 0:

# player.PlayStream(playbackTarget, item, seek, startPlayback=startPlayback, package=package)

#elif float(package.file.resume) > 0:

# player.PlayStream(playbackTarget, item, package.file.resume, startPlayback=startPlayback, package=package)

#else:

# player.PlayStream(playbackTarget, item, 0, startPlayback=startPlayback, package=package)

#loop until finished

while not player.isExit:

player.saveTime()

xbmc.sleep(5000)

elif mode == 'search' and contextType != '':

if title == '':

try:

dialog = xbmcgui.Dialog()

title = dialog.input(addon.getLocalizedString(30110), type=xbmcgui.INPUT\_ALPHANUM)

except:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30100))

title = 'test'

mediaItems = service.getMediaList(title=title, contentType=contentType)

resolvedPlayback = False

startPlayback = False

options = []

urls = []

if mediaItems:

for item in mediaItems:

if item.file is None:

service.addDirectory( item.folder, contextType=contextType)

else:

service.addMediaFile(item, contextType=contextType)

# non-encfs

else:

# file ID provided

#if we don't have the docid, search for the video for playback

if (filename != '' and mode == 'audio'):

mediaFile = file.file(filename, title, '', service.MEDIA\_TYPE\_MUSIC, '','')

mediaFolder = folder.folder(folderID,'')

(mediaURLs,package) = service.getPlaybackCall(package=package.package(mediaFile,mediaFolder))

elif filename != '':

mediaFile = file.file(filename, title, '', 0, '','')

mediaFolder = folder.folder(folderID,'')

(mediaURLs,package) = service.getPlaybackCall(package=package.package(mediaFile,mediaFolder))

# search

elif mode == 'search' and contextType == '':

if title == '':

try:

dialog = xbmcgui.Dialog()

title = dialog.input(addon.getLocalizedString(30110), type=xbmcgui.INPUT\_ALPHANUM)

except:

xbmcgui.Dialog().ok(addon.getLocalizedString(30000), addon.getLocalizedString(30100))

title = 'test'

mediaItems = service.getMediaList(title=title, contentType=contentType)

resolvedPlayback = False

startPlayback = False

options = []

urls = []

if mediaItems:

for item in mediaItems:

if item.file is None:

service.addDirectory( item.folder, contextType=contextType)

else:

options.append(item.file.title)

urls.append(service.addMediaFile(item, contextType=contextType))

#search from STRM

if contextType == '':

ret = xbmcgui.Dialog().select(addon.getLocalizedString(30112), options)

playbackPath = urls[ret]

item = xbmcgui.ListItem(path=playbackPath+'|' + service.getHeadersEncoded())

item.setInfo( type="Video", infoLabels={ "Title": options[ret] , "Plot" : options[ret] } )

xbmcplugin.setResolvedUrl(int(sys.argv[1]), True, item)

# playback of entire folder?

# folder only

elif folderID != '' and title == '':

mediaItems = service.getMediaList(folderName=folderID, contentType=contentType)

if mediaItems:

player.setMedia(mediaItems)

player.playList(service)

resolvedPlayback = False

toExit = True

# title provided

else:

(mediaURLs,package) = service.getPlaybackCall(None,title=title)

#ensure there is something play

if package is not None:

# right-click - download (download only + force)

if not seek > 0 and not (settings.download and not settings.play):

# TESTING

if addon\_parameters.spreadsheet and service.cloudResume == '2':

if service.worksheetID == '':

try:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

spreadsheets = service.gSpreadsheet.getSpreadsheetList()

except:

pass

for title in spreadsheets.iterkeys():

if title == 'CLOUD\_DB':

worksheets = service.gSpreadsheet.getSpreadsheetWorksheets(spreadsheets[title])

for worksheet in worksheets.iterkeys():

if worksheet == 'db':

service.worksheetID = worksheets[worksheet]

addon.setSetting(instanceName + '\_spreadsheet', service.worksheetID)

break

break

# TESTING

if addon\_parameters.spreadsheet and service.cloudResume == '2':

if service.gSpreadsheet is None:

service.gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

media = service.gSpreadsheet.updateMediaPackage(service.worksheetID, package)

if package.file.commands != '':

exp = re.compile('([^\|]+):([^\|]+)\|?', re.IGNORECASE)

for cmd in exp.finditer(package.file.commands):

if cmd.group(1) == 'seek':

seek = cmd.group(2)

elif cmd.group(1) == 'title':

package.file.title = cmd.group(2)

elif cmd.group(1) == 'resume':

package.file.resume = cmd.group(2)

elif cmd.group(1) == 'original':

if cmd.group(2).lower() == 'true':

settings.playOriginal = True

else:

settings.playOriginal = False

elif cmd.group(1) == 'promptquality':

if cmd.group(2).lower() == 'true':

settings.promptQuality = True

else:

settings.promptQuality = False

item = xbmcgui.ListItem(package.file.displayTitle(), iconImage=package.file.thumbnail,

thumbnailImage=package.file.thumbnail)

item.setInfo( type="Video", infoLabels={ "Title": package.file.title , "Plot" : package.file.title } )

originalURL = ''

if mode != 'audio':

cache = cache.cache(package)

service.cache = cache

package.file.thumbnail = cache.setThumbnail(service)

# SRTURL = ''

srtpath = ''

if settings.srt and service.protocol == 2:

cache.setSRT(service)

# download closed-captions

if settings.cc and service.protocol == 2:

cache.setCC(service)

mediaURL = service.getMediaSelection(mediaURLs, folderID, filename)

#mediaURL.url = mediaURL.url +'|' + service.getHeadersEncoded()

# if not seek > 0 and package.file.resume > 0 and not settings.cloudResumePrompt:

# returnPrompt = xbmcgui.Dialog().yesno(addon.getLocalizedString(30000), addon.getLocalizedString(30176), str(int(float(package.file.resume)/360)) + ':'+ str(int(float(package.file.resume)/60)) + ':' + str(int(float(package.file.resume)%60)))

# if not returnPrompt:

# package.file.resume = 0

###

#right-menu context OR STRM

##

if contextType == '':

# right-click - download (download only + force)

if not mediaURL.offline and settings.download and not settings.play:

# service.downloadMediaFile('',playbackPath, str(title)+'.'+ str(playbackQuality), folderID, filename, fileSize, force=True)

service.downloadMediaFile(mediaURL, item, package, force=True, playback=service.PLAYBACK\_NONE)

resolvedPlayback = False

startPlayback = False

# right-click - play + cache (download and play)

elif not mediaURL.offline and settings.download and settings.play:

# service.downloadMediaFile(int(sys.argv[1]), playbackPath, str(title)+'.'+ str(playbackQuality), folderID, filename, fileSize)

service.downloadMediaFile(mediaURL, item, package, playback=service.PLAYBACK\_PLAYER, player=player)

resolvedPlayback = False

# STRM (force resolve) -- resolve-only

elif settings.username != '' or settings.strm:

startPlayback = False

resolvedPlayback = True

startPlayback = False

if not seek > 0 and package.file.cloudResume > 0 and not settings.cloudResumePrompt:

returnPrompt = xbmcgui.Dialog().yesno(addon.getLocalizedString(30000), addon.getLocalizedString(30176), str(int(float(package.file.cloudResume)/360)) + ':'+ str(int(float(package.file.cloudResume)/60)) + ':' + str(int(float(package.file.cloudResume)%60)))

if not returnPrompt:

package.file.resume = 0

else:

package.file.resume = package.file.cloudResume

item.setProperty('isResumable', '1')

item.setProperty('ResumeTime', str(package.file.resume))

item.setProperty('TotalTime', str(package.file.duration))

# right-click - play original / SRT / CC / Start At

elif settings.playOriginal or settings.srt or settings.cc or settings.seek:

startPlayback = True

resolvedPlayback = False

#### not in use

elif 0 and settings.resume:

spreadshetModule = settings.getSetting('library', False)

spreadshetName = settings.getSetting('library\_filename', 'TVShows')

media = {}

if spreadshetModule:

try:

gSpreadsheet = gSpreadsheets.gSpreadsheets(service,addon, user\_agent)

service.gSpreadsheet = gSpreadsheet

spreadsheets = gSpreadsheet.getSpreadsheetList()

except:

spreadshetModule = False

if spreadshetModule:

for title in spreadsheets.iterkeys():

if title == spreadshetName:

worksheets = gSpreadsheet.getSpreadsheetWorksheets(spreadsheets[title])

for worksheet in worksheets.iterkeys():

if worksheet == 'db':

media = gSpreadsheet.getMedia(worksheets[worksheet], fileID=package.file.id)

item = xbmcgui.ListItem(package.file.displayTitle(), iconImage=package.file.thumbnail,

thumbnailImage=package.file.thumbnail)

item.setInfo( type="Video", infoLabels={ "Title": package.file.title , "Plot" : package.file.title } )

player.setWorksheet(worksheets['db'])

if len(media) == 0:

player.PlayStream(mediaURL.url, item, 0, package)

else:

player.PlayStream(mediaURL.url, item,media[0][7],package)

while not player.isExit:

player.saveTime()

xbmc.sleep(5000)

#offline

elif mediaURL.offline:

resolvedPlayback = True

# left-click - always cache (download and play)

elif not mediaURL.offline and settings.download and settings.play:

service.downloadMediaFile(mediaURL, item, package, player=player)

resolvedPlayback = False

else:

resolvedPlayback = True

else:

cache = cache.cache(package)

service.cache = cache

(localResolutions,localFiles) = service.cache.getFiles(service)

if len(localFiles) > 0:

mediaURL = mediaurl.mediaurl(str(localFiles[0]), 'offline', 0, 0)

mediaURL.offline = True

else:

mediaURL = mediaURLs[0]

if not settings.download:

mediaURL.url = mediaURL.url +'|' + service.getHeadersEncoded()

resolvedPlayback = True

###

#right-menu context or STRM

##

if contextType == '':

#download - only, no playback

if not mediaURL.offline and settings.download and not settings.play:

service.downloadMediaFile(mediaURL, item, package, force=True, playback=service.PLAYBACK\_NONE)

resolvedPlayback = False

# for STRM (force resolve) -- resolve-only

elif settings.username != '':

startPlayback = False

#download & playback

elif not mediaURL.offline and settings.download and settings.play:

service.downloadMediaFile(mediaURL, item, package, playback=service.PLAYBACK\_PLAYER, player=player)

resolvedPlayback = False

else:

startPlayback = True

# from within pictures mode, music won't be playable, force

#direct playback from within plugin

elif contextType == 'image' and settings.cache:

item = xbmcgui.ListItem(path=str(playbackPath))

# local, not remote. "Music" is ok

item.setInfo( type="Music", infoLabels={ "Title": title } )

player.play(mediaURL.url, item)

resolvedPlayback = False

# from within pictures mode, music won't be playable, force

#direct playback from within plugin

elif contextType == 'image':

item = xbmcgui.ListItem(package.file.displayTitle(), iconImage=package.file.thumbnail,

thumbnailImage=package.file.thumbnail, path=mediaURL.url)

# for unknown reasons, for remote music, if Music is tagged as Music, it errors-out when playing back from "Music", doesn't happen when labeled "Video"

item.setInfo( type="Video", infoLabels={ "Title": title } )

player.play(mediaURL.url, item)

resolvedPlayback = False

#download and play

elif settings.download and settings.play:

service.downloadMediaFile(mediaURL, item, package, player=player)

resolvedPlayback = False

if float(package.file.cloudResume) > 0 or float(package.file.resume) > 0:

options = []

options.append('Resume from ' + str(int(float(package.file.resume))/60).zfill(2) +':' + str(int(float(package.file.resume))%60).zfill(2) )

options.append('Start from begining')

ret = xbmcgui.Dialog().select(addon.getLocalizedString(30176), options)

if ret == 1:

package.file.resume = 0

if resolvedPlayback:

item.setPath(mediaURL.url)

xbmcplugin.setResolvedUrl(int(sys.argv[1]), True, item)

## contribution by dabinn

# handle situation where playback is skipped to next file, wait for new source to load

if player.isPlaying():

xbmc.sleep(1000)

# need to seek?

if seek > 0:

player.PlayStream(mediaURL.url, item, seek, startPlayback=startPlayback, package=package)

elif float(package.file.cloudResume) > 0:

player.PlayStream(mediaURL.url, item, package.file.cloudResume, startPlayback=startPlayback, package=package)

elif float(package.file.resume) > 0:

player.PlayStream(mediaURL.url, item, package.file.resume, startPlayback=startPlayback, package=package)

else:

player.PlayStream(mediaURL.url, item, 0, startPlayback=startPlayback, package=package)

# load captions

if (settings.srt or settings.cc) and service.protocol == 2:

while not (player.isPlaying()):

xbmc.sleep(1000)

files = cache.getSRT(service)

for file in files:

if file != '':

try:

#file = file.decode('unicode-escape')

file = file.encode('utf-8')

except:

pass

player.setSubtitles(file)

# we need to keep the plugin alive for as long as there is playback from the plugin, or the player object closes

while not player.isExit:

player.saveTime()

xbmc.sleep(5000)

xbmcplugin.endOfDirectory(plugin\_handle)

#automation - create strm files

if service is not None and instanceName is not None and settings.strm:

import time

currentDate = time.strftime("%Y%m%d")

if addon.getSetting(instanceName+'\_changedate') == '' or int(addon.getSetting(instanceName+'\_changedate')) < int(currentDate):

try:

path = settings.getSetting('strm\_path')

except:

pass

if path != '':

try:

pDialog = xbmcgui.DialogProgressBG()

pDialog.create(addon.getLocalizedString(30000), 'Building STRMs...')

except:

pass

#service = gdrive\_api2.gdrive(PLUGIN\_URL,addon,instanceName, user\_agent, settings)

try:

addon.setSetting(instanceName + '\_changedate', currentDate)

service.buildSTRM2(path, contentType=contentType, pDialog=pDialog)

except:

pass

try:

pDialog.update(100)

pDialog.close()

except:

pass

# player = gPlayer.gPlayer()

# player.play(playbackURL+'|' + service.getHeadersEncoded(), item)

# while not (player.isPlaying()):

# xbmc.sleep(1)

# player.seekTime(1000)

# w = tvWindow.tvWindow("tvWindow.xml",addon.getAddonInfo('path'),"Default")

# w.setPlayer(player)

# w.doModal()

# player.seekTime(1000)

# w = tvWindow.tvWindow("tvWindow.xml",addon.getAddonInfo('path'),"Default")

# w.setPlayer(player)

# w.doModal()

# xbmc.executebuiltin("XBMC.PlayMedia("+str(playbackPath)+'|' + service.getHeadersEncoded()+")")

#media = gSpreadsheet.setMediaStatus(worksheets[worksheet], package, watched=2, resume=2)

#item = xbmcgui.ListItem(package.file.displayTitle(), iconImage=package.file.thumbnail,

# thumbnailImage=package.file.thumbnail)

#item.setInfo( type="Video", infoLabels={ "Title": package.file.title , "Plot" : package.file.title } )

#player = gPlayer.gPlayer()

#player.setService(service)

#player.setWorksheet(worksheets['db'])

#if len(media) == 0:

# player.PlayStream(mediaURL.url, item, 0, package)

#else:

# player.PlayStream(mediaURL.url, item,media[0][7],package)

#while not player.isExit:

# player.saveTime()

# xbmc.sleep(5000)