'''

gdrive for KODI / XBMC Plugin

Copyright (C) 2013-12016 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/>.

'''

import os

import sys

import re

import urllib, urllib2

import xbmc, xbmcaddon, xbmcgui, xbmcplugin

import addon\_parameters

class gPlayer(xbmc.Player):

try:

import pysrc.pydevd as pydevd

# stdoutToServer and stderrToServer redirect stdout and stderr to eclipse console

pydevd.settrace('localhost', stdoutToServer=True, stderrToServer=True)

except :

pass

def \_\_init\_\_( self, \*args, \*\*kwargs ):

xbmc.Player.\_\_init\_\_( self )

self.isExit = False

self.seek = 0

self.package = None

self.time = 0

self.service = None

self.current = 1

self.playStatus = False

self.currentURL = ''

def setService(self,service):

self.service = service

def setWorksheet(self,worksheet):

self.worksheet = worksheet

def setContent(self, episodes):

self.content = episodes

self.current = 0

def setMedia(self, mediaItems):

self.mediaItems = mediaItems

self.current = 0

def next(self):

# log('video ' + str(episodes[self.current][CONSTANTS.D\_SOURCE]) + ',' + str(episodes[self.current][CONSTANTS.D\_SHOW]))

# addVideo('plugin://plugin.video.gdrive?mode=playvideo&amp;title='+episodes[video][0],

# { 'title' : str(episodes[video][CONSTANTS.D\_SHOW]) + ' - S' + str(episodes[video][CONSTANTS.D\_SEASON]) + 'xE' + str(episodes[video][CONSTANTS.D\_EPISODE]) + ' ' + str(episodes[video][CONSTANTS.D\_PART]) , 'plot' : episodes[video][CONSTANTS.D\_SHOW] },

# img='None')

# play video

# if self.isExit == 0:

self.play('plugin://plugin.video.gdrive-testing/?mode=video&instance='+str(self.service.instanceName)+'&title='+self.content[self.current][0])

#self.play('plugin://plugin.video.gdrive/?mode=video&instance='+str(self.service.instanceName)+'&title='+self.content[self.current][0])

# self.play(self.content[self.current][0])

# self.tvScheduler.setVideoWatched(self.worksheet, self.content[self.current][0])

# self.tvScheduler.createRow(self.worksheet, '','','','')

if self.current < len(self.content):

self.current += 1

else:

self.current = 0

def saveTime(self):

try:

newTime = self.getTime()

if newTime > self.seek:

self.time = newTime

except:

pass

def PlayStream(self, url, item, seek, startPlayback=True, package=None):

self.currentURL = url

if startPlayback:

self.play(url, item)

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Playback url ' + str(url), xbmc.LOGNOTICE)

if package is not None:

self.package = package

if seek != '':

self.seek = float(seek)

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Seek ' + str(seek), xbmc.LOGNOTICE)

# self.tvScheduler.setVideoWatched(self.worksheet, self.content[self.current][0])

# if seek > 0 and seek !='':

# while not self.isPlaying(): #<== The should be while self.isPlaying():

# print "LOOP"

# xbmc.sleep(500)

# xbmc.sleep(2000)

# print "SEEK "+str(seek)

# self.time = float(seek)

# self.seekTime(float(seek))

def playNext(self, service, package):

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

options = []

mediaURLs = sorted(mediaURLs)

for mediaURL in mediaURLs:

options.append(mediaURL.qualityDesc)

if mediaURL.qualityDesc == 'original':

originalURL = mediaURL.url

playbackURL = ''

playbackQuality = ''

playbackPath = ''

if service.settings.promptQuality:

if len(options) > 1:

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

else:

ret = 0

else:

ret = 0

playbackURL = mediaURLs[ret].url

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Play next ' + str(playbackURL), xbmc.LOGNOTICE)

playbackQuality = mediaURLs[ret].quality

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

thumbnailImage=package.file.thumbnail, path=playbackURL+'|' + service.getHeadersEncoded())

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

self.PlayStream(playbackURL+'|' + service.getHeadersEncoded(),item,0,package)

def playList(self, service):

while self.current < len(self.mediaItems) and not self.isExit:

self.playNext(service, self.mediaItems[self.current])

current = self.current

while current == self.current and not self.isExit:

xbmc.sleep(3000)

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Exit play list', xbmc.LOGNOTICE)

def onPlayBackSeek(self,offset):

print "PLAYBACK SEEKED"

def onPlayBackStarted(self):

print "PLAYBACK STARTED"

self.playStatus = True

#self.tag = xbmc.Player().getVideoInfoTag()

# if self.seek > 0:

# self.seekTime(self.seek)

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Play started', xbmc.LOGNOTICE)

if self.seek > 0 and self.seek !='':

# while not self.isPlaying(): #<== The should be while self.isPlaying():

# print "LOOP"

# xbmc.sleep(500)

# xbmc.sleep(2000)

print "SEEK "+str(self.seek)

self.time = float(self.seek)

self.seekTime(float(self.seek))

self.seek = 0

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Seek time ' + str(self.seek), xbmc.LOGNOTICE)

try:

if (self.service.settings.tv\_watch or self.service.settings.movie\_watch):

result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetEpisodes", "params": { "sort": {"method":"lastplayed"}, "filter": {"field": "title", "operator": "isnot", "value":"1"}, "properties": [ "file"]}, "id": "1"}')

#fixedTitle = re.escape(re.sub('.strm', '',self.package.file.title))

fixedTitle = re.escape(urllib.quote\_plus(self.package.file.title))

#fixedTitle = re.escape(re.sub('[','',fixedTitle))

foundMatch=0

#exp = re.search('"episodeid":(\d+), "file":"[^\"]+"', result)

#for match in re.finditer('"episodeid":(\d+)\,"file"\:".\*'+str(fixedTitle)+'\.strm"', result):#, re.S):

for match in re.finditer('"episodeid":(\d+)\,"file"\:"[^\"]+'+str(fixedTitle)+'.strm"', result):#, re.S):

#for match in re.finditer('"episodeid":(\d+)\,"file"\:"[^\"]+","label"\:"'+str(fixedTitle)+'"', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found show ID '+ match.group(1), xbmc.LOGNOTICE)

self.package.file.TVID = match.group(1)

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

# xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "playcount": '+str(self.package.file.playcount+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

foundMatch=1

break

if foundMatch == 0:

for match in re.finditer('"episodeid":(\d+)\,"file"\:"[^\"]+'+str(re.escape(self.package.file.title))+'.strm"', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found show ID '+ match.group(1), xbmc.LOGNOTICE)

self.package.file.TVID = match.group(1)

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

# xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "playcount": '+str(self.package.file.playcount+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

foundMatch=1

break

if self.service.settings and foundMatch==1:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Found local tv db id='+str(self.package.file.TVID), xbmc.LOGNOTICE)

if foundMatch == 0:

result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetMovies", "params": { "sort": {"method":"lastplayed"}, "filter": {"field": "title", "operator": "isnot", "value":"1"}, "properties": [ "file"]}, "id": "1"}')

for match in re.finditer('"file":"[^\"]+'+str(fixedTitle)+'.strm","label":"[^\"]+","movieid":(\d+)', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found movie ID '+ match.group(1), xbmc.LOGNOTICE)

self.package.file.MOVIEID = match.group(1)

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

#xbmc.executeJSONRPC('{"params": {"movieid": '+str(match.group(1))+', "playcount": '+str(self.package.file.playcount+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

break

if foundMatch == 0:

for match in re.finditer('"file":"[^\"]+'+str(re.escape(self.package.file.title))+'.strm","label":"[^\"]+","movieid":(\d+)', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found movie ID '+ match.group(1), xbmc.LOGNOTICE)

self.package.file.MOVIEID = match.group(1)

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

#xbmc.executeJSONRPC('{"params": {"movieid": '+str(match.group(1))+', "playcount": '+str(self.package.file.playcount+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

break

if self.service.settings and foundMatch==1:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Found local movie db id='+str(self.package.file.MOVIEID), xbmc.LOGNOTICE)

except: pass

def onPlayBackEnded(self):

xbmc.log(self.service.addon.getAddonInfo('name') + ': PLAYBACK ENDED', xbmc.LOGNOTICE)

# self.next()

if self.package is not None:

try:

if addon\_parameters.spreadsheet and self.service.cloudResume == '1' and self.service.protocol == 2 and self.time > self.package.file.resume:

xbmc.log(self.service.addon.getAddonInfo('name') + ': PLAYBACK ENDED 1 ' + str(self.package.file.playcount), xbmc.LOGNOTICE)

self.service.setProperty(self.package.file.id,'resume', self.time)

self.service.setProperty(self.package.file.id,'playcount', int(self.package.file.playcount)+1)

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Updated remote db ', xbmc.LOGNOTICE)

elif addon\_parameters.spreadsheet and self.service.cloudResume == '2' and self.service.protocol == 2 and (self.time/self.package.file.duration) >= int(self.service.settings.skipResume)\*0.01:#and self.time > self.package.file.resume:

xbmc.log(self.service.addon.getAddonInfo('name') + ': PLAYBACK ENDED 2 ' + str(self.package.file.playcount), xbmc.LOGNOTICE)

self.service.gSpreadsheet.setMediaStatus(self.service.worksheetID,self.package, watched= int(self.package.file.playcount)+1, resume=0)

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Updated local db ', xbmc.LOGNOTICE)

# if self.service.settings.tv\_watch:

# exp = re.search('0?(\d+)', self.package.file.season)

# season = exp.group(1)

# exp = re.search('0?(\d+)', self.package.file.episode)

# episode = exp.group(1)

# result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetEpisodes", "params": {"filter":{"and": [{"field": "season", "operator": "is", "value": "'+str(season)+'"}, {"field": "episode", "operator": "is", "value": "'+str(episode)+'"}]}}, "id": 1}')

# exp = re.search('"episodeid":(\d+)', result)

# episodeID = exp.group(1)

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(episodeID)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

# xbmc.executeJSONRPC('{"params": {"episodeid": '+str(episodeID)+', "playcount": '+str(self.package.file.playcount+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

except: pass

try:

if (self.service.settings.tv\_watch and self.package.file.TVID != None):

xbmc.executeJSONRPC('{"params": {"episodeid": '+str(self.package.file.TVID)+', "playcount": '+str(int(self.package.file.playcount)+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

xbmc.log(self.service.addon.getAddonInfo('name') + ': 1 Updated watch status local tv db id='+str(self.package.file.TVID) + ' playcount='+ str(int(self.package.file.playcount)+1), xbmc.LOGNOTICE)

elif (self.service.settings.movie\_watch and self.package.file.MOVIEID != None):

xbmc.executeJSONRPC('{"params": {"movieid": '+str(self.package.file.MOVIEID)+', "playcount": '+str(int(self.package.file.playcount)+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

xbmc.log(self.service.addon.getAddonInfo('name') + ': 1 Updated watch status local movie db id='+str(self.package.file.MOVIEID) + ' playcount='+ str(int(self.package.file.playcount)+1), xbmc.LOGNOTICE)

else:

if (self.service.settings.tv\_watch or self.service.settings.movie\_watch):

result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetEpisodes", "params": { "sort": {"method":"lastplayed"}, "filter": {"field": "title", "operator": "isnot", "value":"1"}, "properties": [ "file"]}, "id": "1"}')

fixedTitle = re.escape(re.sub(' ', '+', self.package.file.title))

foundMatch=0

if (self.service.settings.tv\_watch):

#exp = re.search('"episodeid":(\d+), "file":"[^\"]+"', result)

#for match in re.finditer('"episodeid":(\d+)\,"file"\:".\*'+str(fixedTitle)+'\.strm"', result):#, re.S):

for match in re.finditer('"episodeid":(\d+)\,"file"\:"[^\"]\*'+str(fixedTitle)+'[^\"]\*"', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found show ID '+ match.group(1), xbmc.LOGNOTICE)

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "playcount": '+str(int(self.package.file.playcount)+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

foundMatch=1

break

if self.service.settings and foundMatch==1:

xbmc.log(self.service.addon.getAddonInfo('name') + ': 2 Updated local tv db id='+str(self.package.file.TVID) + ' playcount='+ str(int(self.package.file.playcount)+1), xbmc.LOGNOTICE)

elif self.service.settings.movie\_watch and foundMatch == 0:

result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetMovies", "params": { "sort": {"method":"lastplayed"}, "filter": {"field": "title", "operator": "isnot", "value":"1"}, "properties": [ "file"]}, "id": "1"}')

for match in re.finditer('"file"\:"[^\"]\*'+str(fixedTitle)+'[^\"]\*","label":"[^\"]+","movieid":(\d+)', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found movie ID '+ match.group(1), xbmc.LOGNOTICE)

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

xbmc.executeJSONRPC('{"params": {"movieid": '+str(match.group(1))+', "playcount": '+str(int(self.package.file.playcount)+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

break

if self.service.settings and foundMatch==1:

xbmc.log(self.service.addon.getAddonInfo('name') + ': 2 Updated local movie db id='+str(self.package.file.MOVIEID) + ' playcount='+ str(int(self.package.file.playcount)+1), xbmc.LOGNOTICE)

except: pass

#try:

# self.service.gSpreadsheet.setMediaStatus(self.worksheet,self.package, watched=1)

#except: pass

self.current = self.current +1

self.isExit = True

self.playStatus = False

def onPlayBackStopped(self):

xbmc.log(self.service.addon.getAddonInfo('name') + ': PLAYBACK STOPPED', xbmc.LOGNOTICE)

if self.package is not None:

try:

if addon\_parameters.spreadsheet and self.service.cloudResume == '1' and self.service.protocol == 2 and float(self.time) > float(self.package.file.resume):

xbmc.log(self.service.addon.getAddonInfo('name') + ': PLAYBACK STOPPED 1 ' + str(self.time), xbmc.LOGNOTICE)

self.service.setProperty(self.package.file.id,'resume', self.time)

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Updated remote db ', xbmc.LOGNOTICE)

elif addon\_parameters.spreadsheet and self.service.cloudResume == '2' and self.service.protocol == 2:# and float(self.time) > float(self.package.file.resume):

xbmc.log(self.service.addon.getAddonInfo('name') + ': PLAYBACK STOPPED 2 ' + str(self.time), xbmc.LOGNOTICE)

self.service.gSpreadsheet.setMediaStatus(self.service.worksheetID,self.package, resume=self.time)

# exp = re.search('0?(\d+)', self.package.file.season)

# season = exp.group(1)

# exp = re.search('0?(\d+)', self.package.file.episode)

# episode = exp.group(1)

# result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetEpisodes", "params": {"filter":{"and": [{"field": "season", "operator": "is", "value": "'+str(season)+'"}, {"field": "episode", "operator": "is", "value": "'+str(episode)+'"}]}}, "id": 1}')

# exp = re.search('"episodeid":(\d+)', result)

# episodeID = exp.group(1)

#result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetEpisodes", "params": { "sort": {"method":"lastplayed"}, "filter": {"field": "title", "operator": "isnot", "value":"1"}, "properties": [ "file"], "limits":{"end":3}}, "id": "1"}')

except: pass

try:

if (self.service.settings.tv\_watch or self.service.settings.movie\_watch):

if ( float(self.service.settings.skipResume)/100 < float (self.time/self.package.file.duration)):

if (self.service.settings.tv\_watch and self.package.file.TVID != None):

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(self.package.file.TVID)+', "playcount": '+str(int(self.package.file.playcount)+1)+',"resume": {"position": '+str(1)+', "total": '+str(1)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(self.package.file.TVID)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}, "playcount": '+str(int(self.package.file.playcount)+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(self.package.file.TVID)+', "playcount": '+str(1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

xbmc.executeJSONRPC('{"params": {"episodeid": '+str(self.package.file.TVID)+', "playcount": '+str(int(self.package.file.playcount)+1)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

xbmc.log(self.service.addon.getAddonInfo('name') + ': 1 Updated watch + resume status local tv db id='+str(self.package.file.TVID) + ' playcount='+ str(int(self.package.file.playcount)) + ' time=' +self.time+' duration='+self.package.file.duration, xbmc.LOGNOTICE)

elif (self.service.settings.movie\_watch and self.package.file.MOVIEID != None):

# xbmc.executeJSONRPC('{"params": {"movieid": '+str(self.package.file.MOVIEID)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

xbmc.executeJSONRPC('{"params": {"movieid": '+str(self.package.file.MOVIEID)+', "playcount": '+str(int(self.package.file.playcount)+1)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

xbmc.log(self.service.addon.getAddonInfo('name') + ': 1 Updated watch + resume status local movie db id='+str(self.package.file.MOVIEID) + ' playcount='+ str(int(self.package.file.playcount)) + ' time=' +self.time+' duration='+self.package.file.duration, xbmc.LOGNOTICE)

else:

if (self.service.settings.tv\_watch and self.package.file.TVID != None):

xbmc.executeJSONRPC('{"params": {"episodeid": '+str(self.package.file.TVID)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

xbmc.log(self.service.addon.getAddonInfo('name') + ': 2 Updated resume status local tv db id='+str(self.package.file.TVID) + ' playcount='+ str(int(self.package.file.playcount)) + ' time=' +self.time+' duration='+self.package.file.duration, xbmc.LOGNOTICE)

elif (self.service.settings.movie\_watch and self.package.file.MOVIEID != None):

xbmc.executeJSONRPC('{"params": {"movieid": '+str(self.package.file.MOVIEID)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

xbmc.log(self.service.addon.getAddonInfo('name') + ': 2 Updated resume status local movie db id='+str(self.package.file.MOVIEID) + ' playcount='+ str(int(self.package.file.playcount)) + ' time=' +self.time+' duration='+self.package.file.duration, xbmc.LOGNOTICE)

elif (self.service.settings.tv\_watch or self.service.settings.movie\_watch):

result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetEpisodes", "params": { "sort": {"method":"lastplayed"}, "filter": {"field": "title", "operator": "isnot", "value":"1"}, "properties": [ "file"]}, "id": "1"}')

fixedTitle = re.escape(re.sub(' ', '+', self.package.file.title))

foundMatch=0

if (self.service.settings.tv\_watch):

#exp = re.search('"episodeid":(\d+), "file":"[^\"]+"', result)

#for match in re.finditer('"episodeid":(\d+)\,"file"\:".\*'+str(fixedTitle)+'\.strm"', result):#, re.S):

for match in re.finditer('"episodeid":(\d+)\,"file"\:"[^\"]\*'+str(fixedTitle)+'[^\"]\*"', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found show ID '+ match.group(1), xbmc.LOGNOTICE)

xbmc.executeJSONRPC('{"params": {"episodeid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

foundMatch=1

break

#sxbmc.executeJSONRPC('{"params": {"episodeid": '+str(episodeID)+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

#xbmc.executeJSONRPC('{"params": {"episodeid": '+str(episodeID)+', "playcount": '+str(self.package.file.playcount+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetEpisodeDetails"}')

if self.service.settings and foundMatch ==1:

xbmc.log(self.service.addon.getAddonInfo('name') + ': 3 Updated local tv db playcount='+ str(int(self.package.file.playcount)) + ' time=' +self.time+' duration='+self.package.file.duration, xbmc.LOGNOTICE)

elif self.service.settings.movie\_watch and foundMatch == 0:

result = xbmc.executeJSONRPC('{"jsonrpc": "2.0", "method": "VideoLibrary.GetMovies", "params": { "sort": {"method":"lastplayed"}, "filter": {"field": "title", "operator": "isnot", "value":"1"}, "properties": [ "file"]}, "id": "1"}')

for match in re.finditer('"file"\:"[^\"]\*'+str(fixedTitle)+'[^\"]\*","label":"[^\"]+","movieid":(\d+)', result):#, re.S):

xbmc.log(self.service.addon.getAddonInfo('name') + ': found movie ID '+ match.group(1), xbmc.LOGNOTICE)

xbmc.executeJSONRPC('{"params": {"movieid": '+str(match.group(1))+', "resume": {"position": '+str(self.time)+', "total": '+str(self.package.file.duration)+'}}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

#xbmc.executeJSONRPC('{"params": {"movieid": '+str(match.group(1))+', "playcount": '+str(self.package.file.playcount+1)+'}, "jsonrpc": "2.0", "id": "setResumePoint", "method": "VideoLibrary.SetMovieDetails"}')

break

if self.service.settings and foundMatch==1:

xbmc.log(self.service.addon.getAddonInfo('name') + ': 3 Updated local movie db playcount='+ str(int(self.package.file.playcount)) + ' time=' +self.time+' duration='+self.package.file.duration , xbmc.LOGNOTICE)

except: pass

#self.current = self.current +1

self.isExit = True

# if not self.isExit:

# print "don't exit"

self.playStatus = False

def onPlayBackPaused(self):

print "PLAYBACK Paused"

if self.seek > 0:

self.seekTime(self.seek)

self.seek = 0

def seekTo(self, seek):

if seek != '':

self.seek = float(seek)

# self.tvScheduler.setVideoWatched(self.worksheet, self.content[self.current][0])

if seek > 0 and seek !='':

while not self.isPlaying(): #<== The should be while self.isPlaying():

print "LOOP"

xbmc.sleep(500)

xbmc.sleep(2000)

print "SEEK "+str(seek)

self.time = float(seek)

self.seekTime(float(seek))

if self.service.settings:

xbmc.log(self.service.addon.getAddonInfo('name') + ': Seek ' + str(self.time), xbmc.LOGNOTICE)