############################################################################

# /T /I #

# / |/ | .-~/ #

# T\ Y I |/ / \_ #

# /T | \I | I Y.-~/ #

# I l /I T\ | | l | T / #

# T\ | \ Y l /T | \I l \ ` l Y If your going to copy #

# \_\_ | \l \l \I l \_\_l l \ ` \_. | this addon just #

# \ ~-l `\ `\ \ \ ~\ \ `. .-~ | give credit! #

# \ ~-. "-. ` \ ^.\_ ^. "-. / \ | #

#.--~-.\_ ~- ` \_ ~-\_.-"-." .\_ /.\_ ." ./ Stop Deleting the #

# >--. ~-. .\_ ~>-" "\ 7 7 ] credits file! #

#^.\_\_\_~"--.\_ ~-{ .-~ . `\ Y . / | #

# <\_\_ ~"-. ~ /\_/ \ \I Y : | #

# ^-.\_\_ ~(\_/ \ >.\_: | l\_\_\_\_\_\_ #

# ^--.,\_\_\_.-~" /\_/ ! `-.~"--l\_ / ~"-. #

# (\_/ . ~( /' "~"--,Y -=b-. \_) #

# (\_/ . \ : / l c"~o \ #

# \ / `. . .^ \\_.-~"~--. ) #

# (\_/ . ` / / ! )/ #

# / / \_. '. .': / ' #

# ~(\_/ . / \_ ` .-<\_ #

# /\_/ . ' .-~" `. / \ \ ,z=. Surfacingx #

# ~( / ' : | K "-.~-.\_\_\_\_\_\_// Original Author #

# "-,. l I/ \\_ \_\_{--->.\_(==. #

# //( \ < ~"~" // #

# /' /\ \ \ ,v=. (( Fire TV Guru #

# .^. / /\ " }\_\_ //===- ` PyXBMCt LaYOUt #

# / / ' ' "-.,\_\_ {---(==- #

# .^ ' : T ~" ll #

# / . . . : | :! \ #

# (\_/ / | | j-" ~^ #

# ~-<\_(\_.^-~" #

# #

# Copyright (C) One of those Years.... #

# #

# 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. #

# #

############################################################################

import xbmc, xbmcaddon, xbmcgui, xbmcplugin, os, sys, xbmcvfs, glob

import shutil

import urllib2,urllib

import re

import uservar

import time

from resources.libs import yt, wizard as wiz

from datetime import date, datetime, timedelta

ADDON\_ID = uservar.ADDON\_ID

ADDON = wiz.addonId(ADDON\_ID)

VERSION = wiz.addonInfo(ADDON\_ID,'version')

ADDONPATH = wiz.addonInfo(ADDON\_ID,'path')

ADDONTITLE = uservar.ADDONTITLE

BUILDERNAME = uservar.BUILDERNAME

DIALOG = xbmcgui.Dialog()

DP = xbmcgui.DialogProgress()

HOME = xbmc.translatePath('special://home/')

ADDONS = os.path.join(HOME, 'addons')

USERDATA = os.path.join(HOME, 'userdata')

PLUGIN = os.path.join(ADDONS, ADDON\_ID)

PACKAGES = os.path.join(ADDONS, 'packages')

ADDONDATA = os.path.join(USERDATA, 'addon\_data', ADDON\_ID)

FANART = os.path.join(ADDONPATH, 'fanart.jpg')

ICON = os.path.join(ADDONPATH, 'icon.png')

ART = os.path.join(ADDONPATH, 'resources', 'art')

SKINFOLD = os.path.join(ADDONPATH, 'resources', 'skins', 'DefaultSkin', 'media')

ADVANCED = os.path.join(USERDATA, 'advancedsettings.xml')

NOTIFY = wiz.getS('notify')

NOTEID = wiz.getS('noteid')

NOTEDISMISS = wiz.getS('notedismiss')

BUILDNAME = wiz.getS('buildname')

BUILDVERSION = wiz.getS('buildversion')

LATESTVERSION = wiz.checkBuild(BUILDNAME, 'version')

TODAY = date.today()

KODIV = float(xbmc.getInfoLabel("System.BuildVersion")[:4])

TOMORROW = TODAY + timedelta(days=1)

THREEDAYS = TODAY + timedelta(days=3)

UPDATECHECK = uservar.UPDATECHECK if str(uservar.UPDATECHECK).isdigit() else 1

NEXTCHECK = TODAY + timedelta(days=UPDATECHECK)

NOTIFICATION = uservar.NOTIFICATION

ENABLE = uservar.ENABLE

HEADERTYPE = uservar.HEADERTYPE if uservar.HEADERTYPE == 'Image' else 'Text'

HEADERMESSAGE = uservar.HEADERMESSAGE

BACKGROUND = uservar.BACKGROUND

HEADERIMAGE = uservar.HEADERIMAGE

THEME1 = uservar.THEME1

THEME2 = uservar.THEME2

THEME3 = uservar.THEME3

THEME4 = uservar.THEME4

THEME5 = uservar.THEME5

COLOR1 = uservar.COLOR1

COLOR2 = uservar.COLOR2

COLOR3 = uservar.COLOR3

COLOR4 = uservar.COLOR4

COLOR5 = uservar.COLOR5

CONTACTICON = uservar.CONTACTICON if not uservar.CONTACTICON == 'http://' else ICON

CONTACTFANART = uservar.CONTACTFANART if not uservar.CONTACTFANART == 'http://' else FANART

if BACKGROUND == '': BACKGROUND = FANART

elif not wiz.workingURL(BACKGROUND): BACKGROUND = FANART

ACTION\_PREVIOUS\_MENU = 10 ## ESC action

ACTION\_NAV\_BACK = 92 ## Backspace action

ACTION\_MOVE\_LEFT = 1 ## Left arrow key

ACTION\_MOVE\_RIGHT = 2 ## Right arrow key

ACTION\_MOVE\_UP = 3 ## Up arrow key

ACTION\_MOVE\_DOWN = 4 ## Down arrow key

ACTION\_MOUSE\_WHEEL\_UP = 104 ## Mouse wheel up

ACTION\_MOUSE\_WHEEL\_DOWN = 105 ## Mouse wheel down

ACTION\_MOVE\_MOUSE = 107 ## Down arrow key

ACTION\_SELECT\_ITEM = 7 ## Number Pad Enter

ACTION\_BACKSPACE = 110 ## ?

ACTION\_MOUSE\_LEFT\_CLICK = 100

ACTION\_MOUSE\_LONG\_CLICK = 108

def artwork(file):

if file == 'button': return os.path.join(SKINFOLD, 'Button', 'button-focus\_lightblue.png'), os.path.join(SKINFOLD, 'Button', 'button-focus\_grey.png')

elif file == 'radio' : return os.path.join(SKINFOLD, 'RadioButton', 'MenuItemFO.png'), os.path.join(SKINFOLD, 'RadioButton', 'MenuItemNF.png'), os.path.join(SKINFOLD, 'RadioButton', 'radiobutton-focus.png'), os.path.join(SKINFOLD, 'RadioButton', 'radiobutton-nofocus.png')

elif file == 'slider': return os.path.join(SKINFOLD, 'Slider', 'osd\_slider\_nib.png'), os.path.join(SKINFOLD, 'Slider', 'osd\_slider\_nibNF.png'), os.path.join(SKINFOLD, 'Slider', 'slider1.png'), os.path.join(SKINFOLD, 'Slider', 'slider1.png')

def autoConfig(msg='', TxtColor='0xFFFFFFFF', Font='font10', BorderWidth=10):

class MyWindow(xbmcgui.WindowDialog):

scr={};

def \_\_init\_\_(self,msg='',L=0,T=0,W=1280,H=720,TxtColor='0xFFFFFFFF',Font='font10',BorderWidth=10):

buttonfocus, buttonnofocus = artwork('button')

radiobgfocus, radiobgnofocus, radiofocus, radionofocus = artwork('radio')

slidernibfocus, slidernibnofocus, sliderfocus, slidernofocus = artwork('slider')

image\_path = os.path.join(ART, 'ContentPanel.png')

boxbg = os.path.join(ART, 'bgg2.png')

self.border = xbmcgui.ControlImage(L,T,W,H, image\_path)

self.addControl(self.border);

self.BG=xbmcgui.ControlImage(L+BorderWidth,T+BorderWidth,W-(BorderWidth\*2),H-(BorderWidth\*2), FANART, aspectRatio=0, colorDiffuse='0x5FFFFFFF')

self.addControl(self.BG)

top = T+BorderWidth

leftside = L+BorderWidth

rightside = L+(W/2)-(BorderWidth\*2)

firstrow = top+30

secondrow = firstrow+275+(BorderWidth/2)

currentwidth = ((W/2)-(BorderWidth\*4))/2

header = '[COLOR %s]Advanced Settings Configurator[/COLOR]' % (COLOR2)

self.Header=xbmcgui.ControlLabel(L, top, W, 30, header, font='font13', textColor=TxtColor, alignment=0x00000002)

self.addControl(self.Header)

top += 30+BorderWidth

self.bgarea = xbmcgui.ControlImage(leftside, firstrow, rightside-L, 275, boxbg, aspectRatio=0, colorDiffuse='0x5FFFFFFF')

self.addControl(self.bgarea)

self.bgarea2 = xbmcgui.ControlImage(rightside+BorderWidth+BorderWidth, firstrow, rightside-L, 275, boxbg, aspectRatio=0, colorDiffuse='0x5FFFFFFF')

self.addControl(self.bgarea2)

self.bgarea3 = xbmcgui.ControlImage(leftside, secondrow, rightside-L, 275, boxbg, aspectRatio=0, colorDiffuse='0x5FFFFFFF')

self.addControl(self.bgarea3)

self.bgarea4 = xbmcgui.ControlImage(rightside+BorderWidth+BorderWidth, secondrow, rightside-L, 275, boxbg, aspectRatio=0, colorDiffuse='0x5FFFFFFF')

self.addControl(self.bgarea4)

header = '[COLOR %s]Video Cache Size[/COLOR]' % (COLOR2)

self.Header2=xbmcgui.ControlLabel(leftside+BorderWidth, firstrow+5, (W/2)-(BorderWidth\*2), 20, header, font='font13', textColor=TxtColor, alignment=0x00000002)

self.addControl(self.Header2)

freeMemory = int(float(wiz.getInfo('System.Memory(free)')[:-2])\*.33)

recMemory = int(float(wiz.getInfo('System.Memory(free)')[:-2])\*.23)

msg3 = "[COLOR %s]Number of bytes used for buffering streams in memory. When set to [COLOR %s]0[/COLOR] the cache will be written to disk instead of RAM. Note: For the memory size set here, Kodi will require 3x the amount of RAM to be free. Setting this too high might cause Kodi to crash if it can't get enough RAM(1/3 of Free Memory: [COLOR %s]%s[/COLOR])[/COLOR]" % (COLOR2, COLOR1, COLOR1, freeMemory)

self.Support3=xbmcgui.ControlTextBox(leftside+int(BorderWidth\*1.5), firstrow+30+BorderWidth, (W/2)-(BorderWidth\*4), 150, font='font12', textColor=TxtColor)

self.addControl(self.Support3)

self.Support3.setText(msg3)

try: self.videoCacheSize=xbmcgui.ControlSlider(leftside+int(BorderWidth\*1.5), firstrow+210,(W/2)-(BorderWidth\*5),20, textureback=sliderfocus, texture=slidernibnofocus, texturefocus=slidernibfocus, orientation=xbmcgui.HORIZONTAL)

except: self.videoCacheSize=xbmcgui.ControlSlider(leftside+int(BorderWidth\*1.5), firstrow+210,(W/2)-(BorderWidth\*5),20, textureback=sliderfocus, texture=slidernibnofocus, texturefocus=slidernibfocus)

self.addControl(self.videoCacheSize)

self.videomin = 0; self.videomax = freeMemory if freeMemory < 2000 else 2000

self.recommendedVideo = recMemory if recMemory < 500 else 500; self.currentVideo = self.recommendedVideo

videopos = wiz.percentage(self.currentVideo, self.videomax)

self.videoCacheSize.setPercent(videopos)

current1 = '[COLOR %s]Current:[/COLOR] [COLOR %s]%s MB[/COLOR]' % (COLOR1, COLOR2, self.currentVideo)

recommended1 = '[COLOR %s]Recommended:[/COLOR] [COLOR %s]%s MB[/COLOR]' % (COLOR1, COLOR2, self.recommendedVideo)

self.currentVideo1=xbmcgui.ControlTextBox(leftside+BorderWidth,firstrow+235,currentwidth,20,font=Font,textColor=TxtColor)

self.addControl(self.currentVideo1)

self.currentVideo1.setText(current1)

self.recommendedVideo1=xbmcgui.ControlTextBox(leftside+BorderWidth+currentwidth,firstrow+235,currentwidth,20,font=Font,textColor=TxtColor)

self.addControl(self.recommendedVideo1)

self.recommendedVideo1.setText(recommended1)

header = '[COLOR %s]CURL Timeout/CURL Low Speed[/COLOR]' % (COLOR2)

self.Header3=xbmcgui.ControlLabel(rightside+BorderWidth, firstrow+5, (W/2)-(BorderWidth\*2), 20, header, font='font13', textColor=TxtColor, alignment=0x00000002)

self.addControl(self.Header3)

msg3 = "[COLOR %s][B]curlclienttimeout[/B] is the time in seconds for how long it takes for libcurl connection will timeout and [B]curllowspeedtime[/B] is the time in seconds for libcurl to consider a connection lowspeed. For slower connections set it to 20.[/COLOR]" % COLOR2

self.Support3=xbmcgui.ControlTextBox(rightside+int(BorderWidth\*3.5), firstrow+30+BorderWidth, (W/2)-(BorderWidth\*4), 150, font='font12', textColor=TxtColor)

self.addControl(self.Support3)

self.Support3.setText(msg3)

try: self.CURLTimeout=xbmcgui.ControlSlider(rightside+int(BorderWidth\*3.5),firstrow+210,(W/2)-(BorderWidth\*5),20, textureback=sliderfocus, texture=slidernibnofocus, texturefocus=slidernibfocus, orientation=xbmcgui.HORIZONTAL)

except: self.CURLTimeout=xbmcgui.ControlSlider(rightside+int(BorderWidth\*3.5),firstrow+210,(W/2)-(BorderWidth\*5),20, textureback=sliderfocus, texture=slidernibnofocus, texturefocus=slidernibfocus)

self.addControl(self.CURLTimeout)

self.curlmin = 0; self.curlmax = 20

self.recommendedCurl = 10; self.currentCurl = self.recommendedCurl

curlpos = wiz.percentage(self.currentCurl, self.curlmax)

self.CURLTimeout.setPercent(curlpos)

current2 = '[COLOR %s]Current:[/COLOR] [COLOR %s]%ss[/COLOR]' % (COLOR1, COLOR2, self.currentCurl)

recommended2 = '[COLOR %s]Recommended:[/COLOR] [COLOR %s]%ss[/COLOR]' % (COLOR1, COLOR2, self.recommendedCurl)

self.currentCurl2=xbmcgui.ControlTextBox(rightside+(BorderWidth\*3),firstrow+235,currentwidth,20,font=Font,textColor=TxtColor)

self.addControl(self.currentCurl2)

self.currentCurl2.setText(current2)

self.recommendedCurl2=xbmcgui.ControlTextBox(rightside+(BorderWidth\*3)+currentwidth,firstrow+235,currentwidth,20,font=Font,textColor=TxtColor)

self.addControl(self.recommendedCurl2)

self.recommendedCurl2.setText(recommended2)

header = '[COLOR %s]Read Buffer Factor[/COLOR]' % (COLOR2)

self.Header4=xbmcgui.ControlLabel(leftside, secondrow+5, (W/2)-(BorderWidth\*2), 20, header, font='font13', textColor=TxtColor, alignment=0x00000002)

self.addControl(self.Header4)

msg3 = "[COLOR %s]The value of this setting is a multiplier of the default limit. If Kodi is loading a typical bluray raw file at 36 Mbit/s, then a value of 2 will need at least 72 Mbit/s of network bandwidth. However, unlike with the RAM setting, you can safely increase this value however high you want, and Kodi won't crash.[/COLOR]" % COLOR2

self.Support3=xbmcgui.ControlTextBox(leftside+int(BorderWidth\*1.5), secondrow+30+BorderWidth, (W/2)-(BorderWidth\*4), 150, font='font12', textColor=TxtColor)

self.addControl(self.Support3)

self.Support3.setText(msg3)

try: self.readBufferFactor=xbmcgui.ControlSlider(leftside+int(BorderWidth\*1.5), secondrow+210,(W/2)-(BorderWidth\*5),20, textureback=sliderfocus, texture=slidernibnofocus, texturefocus=slidernibfocus, orientation=xbmcgui.HORIZONTAL)

except: self.readBufferFactor=xbmcgui.ControlSlider(leftside+int(BorderWidth\*1.5), secondrow+210,(W/2)-(BorderWidth\*5),20, textureback=sliderfocus, texture=slidernibnofocus, texturefocus=slidernibfocus)

self.addControl(self.readBufferFactor)

self.readmin = 0; self.readmax = 10

self.recommendedRead = 5; self.currentRead = self.recommendedRead

readpos = wiz.percentage(self.currentRead, self.readmax)

self.readBufferFactor.setPercent(readpos)

current3 = '[COLOR %s]Current:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, self.currentRead)

recommended3 = '[COLOR %s]Recommended:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, self.recommendedRead)

self.currentRead3=xbmcgui.ControlTextBox(leftside+BorderWidth,secondrow+235,currentwidth,20,font=Font,textColor=TxtColor)

self.addControl(self.currentRead3)

self.currentRead3.setText(current3)

self.recommendedRead3=xbmcgui.ControlTextBox(leftside+BorderWidth+currentwidth,secondrow+235,currentwidth,20,font=Font,textColor=TxtColor)

self.addControl(self.recommendedRead3)

self.recommendedRead3.setText(recommended3)

header = '[COLOR %s]Buffer Mode[/COLOR]' % (COLOR2)

self.Header4=xbmcgui.ControlLabel(rightside+BorderWidth, secondrow+5, (W/2)-(BorderWidth\*2), 20, header, font='font13', textColor=TxtColor, alignment=0x00000002)

self.addControl(self.Header4)

msg4 = "[COLOR %s]This setting will force Kodi to use a cache for all video files, including local network, internet, and even the local hard drive. Default value is 0 and will only cache videos that use internet file paths/sources.[/COLOR]" % COLOR2

self.Support4=xbmcgui.ControlTextBox(rightside+int(BorderWidth\*3.5), secondrow+30+BorderWidth, (W/2)-(BorderWidth\*4), 110, font='font12', textColor=TxtColor)

self.addControl(self.Support4)

self.Support4.setText(msg4)

B1 = secondrow+130+BorderWidth; B2 = B1+30; B3 = B2+30; B4 = B3+30;

self.Button0 = xbmcgui.ControlRadioButton(rightside+(BorderWidth\*3), B1, (W/2)-(BorderWidth\*4), 30, '0: Buffer all internet filesystems', font='font10', focusTexture=radiobgfocus, noFocusTexture=radiobgnofocus, focusOnTexture=radiofocus, noFocusOnTexture=radiofocus, focusOffTexture=radionofocus, noFocusOffTexture=radionofocus)

self.Button1 = xbmcgui.ControlRadioButton(rightside+(BorderWidth\*3), B2, (W/2)-(BorderWidth\*4), 30, '1: Buffer all filesystems', font='font10', focusTexture=radiobgfocus, noFocusTexture=radiobgnofocus, focusOnTexture=radiofocus, noFocusOnTexture=radiofocus, focusOffTexture=radionofocus, noFocusOffTexture=radionofocus)

self.Button2 = xbmcgui.ControlRadioButton(rightside+(BorderWidth\*3), B3, (W/2)-(BorderWidth\*4), 30, '2: Only buffer true internet filesystems', font='font10', focusTexture=radiobgfocus, noFocusTexture=radiobgnofocus, focusOnTexture=radiofocus, noFocusOnTexture=radiofocus, focusOffTexture=radionofocus, noFocusOffTexture=radionofocus)

self.Button3 = xbmcgui.ControlRadioButton(rightside+(BorderWidth\*3), B4, (W/2)-(BorderWidth\*4), 30, '3: No Buffer', font='font10', focusTexture=radiobgfocus, noFocusTexture=radiobgnofocus, focusOnTexture=radiofocus, noFocusOnTexture=radiofocus, focusOffTexture=radionofocus, noFocusOffTexture=radionofocus)

self.addControl(self.Button0)

self.addControl(self.Button1)

self.addControl(self.Button2)

self.addControl(self.Button3)

self.Button0.setSelected(False)

self.Button1.setSelected(False)

self.Button2.setSelected(True)

self.Button3.setSelected(False)

self.buttonWrite=xbmcgui.ControlButton(leftside,T+H-40-BorderWidth,(W/2)-(BorderWidth\*2),35,"Write File",textColor="0xFF000000",focusedColor="0xFF000000",alignment=2,focusTexture=buttonfocus,noFocusTexture=buttonnofocus)

self.buttonCancel=xbmcgui.ControlButton(rightside+BorderWidth\*2,T+H-40-BorderWidth,(W/2)-(BorderWidth\*2),35,"Cancel",textColor="0xFF000000",focusedColor="0xFF000000",alignment=2,focusTexture=buttonfocus,noFocusTexture=buttonnofocus)

self.addControl(self.buttonWrite); self.addControl(self.buttonCancel)

self.buttonWrite.controlLeft(self.buttonCancel); self.buttonWrite.controlRight(self.buttonCancel); self.buttonWrite.controlUp(self.Button3); self.buttonWrite.controlDown(self.videoCacheSize)

self.buttonCancel.controlLeft(self.buttonWrite); self.buttonCancel.controlRight(self.buttonWrite); self.buttonCancel.controlUp(self.Button3); self.buttonCancel.controlDown(self.videoCacheSize)

self.videoCacheSize.controlUp(self.buttonWrite); self.videoCacheSize.controlDown(self.CURLTimeout)

self.CURLTimeout.controlUp(self.videoCacheSize); self.CURLTimeout.controlDown(self.readBufferFactor)

self.readBufferFactor.controlUp(self.CURLTimeout); self.readBufferFactor.controlDown(self.Button0)

self.Button0.controlUp(self.CURLTimeout); self.Button0.controlDown(self.Button1); self.Button0.controlLeft(self.readBufferFactor); self.Button0.controlRight(self.readBufferFactor);

self.Button1.controlUp(self.Button0); self.Button1.controlDown(self.Button2); self.Button1.controlLeft(self.readBufferFactor); self.Button1.controlRight(self.readBufferFactor);

self.Button2.controlUp(self.Button1); self.Button2.controlDown(self.Button3); self.Button2.controlLeft(self.readBufferFactor); self.Button2.controlRight(self.readBufferFactor);

self.Button3.controlUp(self.Button2); self.Button3.controlDown(self.buttonWrite); self.Button3.controlLeft(self.readBufferFactor); self.Button3.controlRight(self.readBufferFactor);

self.setFocus(self.videoCacheSize)

def doExit(self):

self.CloseWindow()

def updateCurrent(self, control):

if control == self.videoCacheSize:

self.currentVideo = (self.videomax)\*self.videoCacheSize.getPercent()/100

current = '[COLOR %s]Current:[/COLOR] [COLOR %s]%s MB[/COLOR]' % (COLOR1, COLOR2, int(self.currentVideo))

self.currentVideo1.setText(current)

elif control == self.CURLTimeout:

self.currentCurl = (self.curlmax)\*self.CURLTimeout.getPercent()/100

current = '[COLOR %s]Current:[/COLOR] [COLOR %s]%ss[/COLOR]' % (COLOR1, COLOR2, int(self.currentCurl))

self.currentCurl2.setText(current)

elif control == self.readBufferFactor:

self.currentRead = (self.readmax)\*self.readBufferFactor.getPercent()/100

current = '[COLOR %s]Current:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, int(self.currentRead))

self.currentRead3.setText(current)

elif control in [self.Button0, self.Button1, self.Button2, self.Button3]:

self.Button0.setSelected(False)

self.Button1.setSelected(False)

self.Button2.setSelected(False)

self.Button3.setSelected(False)

control.setSelected(True)

def doWrite(self):

#self.currentVideo = int((self.videomax-20)\*self.videoCacheSize.getPercent()/100+20)\*1024\*1024

#self.currentCurl = int((self.curlmax)\*self.CURLTimeout.getPercent()/100)

#self.currentRead = int((self.readmax)\*self.readBufferFactor.getPercent()/100)

if self.Button0.isSelected(): buffermode = 0

elif self.Button1.isSelected(): buffermode = 1

elif self.Button2.isSelected(): buffermode = 2

elif self.Button3.isSelected(): buffermode = 3

if os.path.exists(ADVANCED):

choice = DIALOG.yesno(ADDONTITLE, "[COLOR %s]There is currently an active [COLOR %s]AdvancedSettings.xml[/COLOR], would you like to remove it and continue?[/COLOR]" % (COLOR2, COLOR1), yeslabel="[B][COLOR green]Remove Settings[/COLOR][/B]", nolabel="[B][COLOR red]Cancel Write[/COLOR][/B]")

if choice == 0: return

try: os.remove(ADVANCED)

except: f = open(ADVANCED, 'w'); f.close()

if KODIV < 17:

with open(ADVANCED, 'w+') as f:

f.write('<advancedsettings>\n')

f.write(' <network>\n')

f.write(' <buffermode>%s</buffermode>\n' % buffermode)

f.write(' <cachemembuffersize>%s</cachemembuffersize>\n' % int(self.currentVideo\*1024\*1024))

f.write(' <readbufferfactor>%s</readbufferfactor>\n' % self.currentRead)

f.write(' <curlclienttimeout>%s</curlclienttimeout>\n' % self.currentCurl)

f.write(' <curllowspeedtime>%s</curllowspeedtime>\n' % self.currentCurl)

f.write(' </network>\n')

f.write('</advancedsettings>\n')

f.close()

else:

with open(ADVANCED, 'w+') as f:

f.write('<advancedsettings>\n')

f.write(' <cache>\n')

f.write(' <buffermode>%s</buffermode>\n' % buffermode)

f.write(' <memorysize>%s</memorysize>\n' % int(self.currentVideo\*1024\*1024))

f.write(' <readfactor>%s</readfactor>\n' % self.currentRead)

f.write(' </cache>\n')

f.write(' <network>\n')

f.write(' <curlclienttimeout>%s</curlclienttimeout>\n' % self.currentCurl)

f.write(' <curllowspeedtime>%s</curllowspeedtime>\n' % self.currentCurl)

f.write(' </network>\n')

f.write('</advancedsettings>\n')

f.close()

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]AdvancedSettings.xml have been written[/COLOR]' % COLOR2)

self.CloseWindow()

def onControl(self, control):

if control==self.buttonWrite: self.doWrite()

elif control==self.buttonCancel: self.doExit()

def onAction(self, action):

try: F=self.getFocus()

except: F=False

if F == self.videoCacheSize: self.updateCurrent(self.videoCacheSize)

elif F == self.CURLTimeout: self.updateCurrent(self.CURLTimeout)

elif F == self.readBufferFactor: self.updateCurrent(self.readBufferFactor)

elif F in [self.Button0, self.Button1, self.Button2, self.Button3] and action in [ACTION\_MOUSE\_LEFT\_CLICK, ACTION\_SELECT\_ITEM]: self.updateCurrent(F)

elif action == ACTION\_PREVIOUS\_MENU: self.doExit()

elif action == ACTION\_NAV\_BACK: self.doExit()

def CloseWindow(self): self.close()

maxW=1280; maxH=720; W=int(900); H=int(650); L=int((maxW-W)/2); T=int((maxH-H)/2);

TempWindow=MyWindow(L=L,T=T,W=W,H=H,TxtColor=TxtColor,Font=Font,BorderWidth=BorderWidth);

TempWindow.doModal()

del TempWindow

##########################################

# `7MM"""YMM MMP""MM""YMM .g8"""bgd #

# MM `7 P' MM `7 .dP' `M #

# MM d MM dM' ` #

# MM""MM MM MM #

# MM Y MM MM. `7MMF' #

# MM MM `Mb. MM #

# .JMML. .JMML. `"bmmmdPY #

##########################################

def autoConfig2(msg='', TxtColor='0xFFFFFFFF', Font='font10', BorderWidth=10):

class MyWindow(xbmcgui.WindowDialog):

scr={};

def \_\_init\_\_(self,msg='',L=0,T=0,W=1280,H=720,TxtColor='0xFFFFFFFF',Font='font10',BorderWidth=10):

buttonfocus, buttonnofocus = artwork('button')

self.BG=xbmcgui.ControlImage(L+BorderWidth,T+BorderWidth,W-(BorderWidth\*2),H-(BorderWidth\*2), FANART, aspectRatio=0)

self.addControl(self.BG)

top = T+BorderWidth

leftside = L+BorderWidth

rightside = L+(W/2)-(BorderWidth\*2)

header = '[COLOR %s]Quick Advanced Settings Configurator[/COLOR]' % (COLOR3)

self.Header=xbmcgui.ControlLabel(L, top, W, 30, header, font='font13', textColor=TxtColor, alignment=0x00000002)

self.addControl(self.Header)

top += 30+BorderWidth

#####Video Cache Size####

freeMemory = int(float(wiz.getInfo('System.Memory(free)')[:-2])\*.33)

recMemory = int(float(wiz.getInfo('System.Memory(free)')[:-2])\*.23)

self.videomin = 0; self.videomax = freeMemory if freeMemory < 2000 else 2000

self.recommendedVideo = recMemory if recMemory < 500 else 500; self.currentVideo = self.recommendedVideo

current1 = '[COLOR %s]Video Cache Size[/COLOR]=[COLOR %s]%s MB[/COLOR]' % (COLOR1, COLOR2, self.currentVideo)

recommended1 = '[COLOR %s]Video Cache Size:[/COLOR] [COLOR %s]%s MB[/COLOR]' % (COLOR1, COLOR5, self.recommendedVideo)

####CURL Timeout/CURL Low Speed####

self.curlmin = 0; self.curlmax = 20

self.recommendedCurl = 10; self.currentCurl = self.recommendedCurl

curlpos = wiz.percentage(self.currentCurl, self.curlmax)

recommended2 = '[COLOR %s]CURL Timeout/CURL Low Speed:[/COLOR] [COLOR %s]%ss[/COLOR]' % (COLOR1, COLOR5, self.recommendedCurl)

########Read Buffer Factor#####

self.readmin = 0; self.readmax = 10

self.recommendedRead = 5; self.currentRead = self.recommendedRead

readpos = wiz.percentage(self.currentRead, self.readmax)

recommended3 = '[COLOR %s]Read Buffer Factor:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR5, self.recommendedRead)

######Buffer Mode#####

recommended4 = '[COLOR %s]Buffer Mode:[/COLOR] [COLOR %s]2[/COLOR]' %(COLOR1, COLOR5)

####BOX##

msgbox='[COLOR %s]These settings will be written to the advancesettings.xml[/COLOR]\r\n\r\n%s\r\n%s\r\n%s\r\n%s' %(COLOR4, recommended4, recommended1, recommended3, recommended2)

self.box=xbmcgui.ControlTextBox(L+25,T+50,W,H, font='font14')

self.addControl(self.box)

self.box.setText(msgbox)

####Buttons###

self.buttonWrite=xbmcgui.ControlButton(leftside,T+H-40-BorderWidth,(W/2)-(BorderWidth\*2),35,"Write File",textColor="0xFF000000",focusedColor="0xFF000000",alignment=2,focusTexture=buttonfocus,noFocusTexture=buttonnofocus)

self.buttonCancel=xbmcgui.ControlButton(rightside+BorderWidth\*2,T+H-40-BorderWidth,(W/2)-(BorderWidth\*2),35,"Cancel",textColor="0xFF000000",focusedColor="0xFF000000",alignment=2,focusTexture=buttonfocus,noFocusTexture=buttonnofocus)

self.addControl(self.buttonWrite); self.addControl(self.buttonCancel)

self.setFocus(self.buttonCancel)

self.buttonWrite.controlLeft(self.buttonCancel); self.buttonWrite.controlRight(self.buttonCancel); self.buttonCancel.controlLeft(self.buttonWrite); self.buttonCancel.controlRight(self.buttonWrite)

def doExit(self):

self.CloseWindow()

def updateCurrent(self, control):

if control == self.videoCacheSize:

self.currentVideo = (self.videomax)\*self.videoCacheSize.getPercent()/100

current = '[COLOR %s]Current:[/COLOR] [COLOR %s]%s MB[/COLOR]' % (COLOR1, COLOR2, int(self.currentVideo))

self.currentVideo1.setText(current)

elif control == self.CURLTimeout:

self.currentCurl = (self.curlmax)\*self.CURLTimeout.getPercent()/100

current = '[COLOR %s]Current:[/COLOR] [COLOR %s]%ss[/COLOR]' % (COLOR1, COLOR2, int(self.currentCurl))

self.currentCurl2.setText(current)

elif control == self.readBufferFactor:

self.currentRead = (self.readmax)\*self.readBufferFactor.getPercent()/100

current = '[COLOR %s]Current:[/COLOR] [COLOR %s]%s[/COLOR]' % (COLOR1, COLOR2, int(self.currentRead))

self.currentRead3.setText(current)

def doWrite(self):

buffermode = 2

if os.path.exists(ADVANCED):

choice = DIALOG.yesno(ADDONTITLE, "[COLOR %s]There is currently an active [COLOR %s]AdvancedSettings.xml[/COLOR], would you like to remove it and continue?[/COLOR]" % (COLOR2, COLOR1), yeslabel="[B][COLOR green]Remove Settings[/COLOR][/B]", nolabel="[B][COLOR red]Cancel Write[/COLOR][/B]")

if choice == 0: return

try: os.remove(ADVANCED)

except: f = open(ADVANCED, 'w'); f.close()

if KODIV < 17:

with open(ADVANCED, 'w+') as f:

f.write('<advancedsettings>\n')

f.write(' <network>\n')

f.write(' <buffermode>%s</buffermode>\n' % buffermode)

f.write(' <cachemembuffersize>%s</cachemembuffersize>\n' % int(self.currentVideo\*1024\*1024))

f.write(' <readbufferfactor>%s</readbufferfactor>\n' % self.currentRead)

f.write(' <curlclienttimeout>%s</curlclienttimeout>\n' % self.currentCurl)

f.write(' <curllowspeedtime>%s</curllowspeedtime>\n' % self.currentCurl)

f.write(' </network>\n')

f.write('</advancedsettings>\n')

f.close()

else:

with open(ADVANCED, 'w+') as f:

f.write('<advancedsettings>\n')

f.write(' <cache>\n')

f.write(' <buffermode>%s</buffermode>\n' % buffermode)

f.write(' <memorysize>%s</memorysize>\n' % int(self.currentVideo\*1024\*1024))

f.write(' <readfactor>%s</readfactor>\n' % self.currentRead)

f.write(' </cache>\n')

f.write(' <network>\n')

f.write(' <curlclienttimeout>%s</curlclienttimeout>\n' % self.currentCurl)

f.write(' <curllowspeedtime>%s</curllowspeedtime>\n' % self.currentCurl)

f.write(' </network>\n')

f.write('</advancedsettings>\n')

f.close()

wiz.LogNotify("[COLOR %s]%s[/COLOR]" % (COLOR1, ADDONTITLE), '[COLOR %s]AdvancedSettings.xml have been written[/COLOR]' % COLOR2)

self.CloseWindow()

def onControl(self, control):

if control==self.buttonWrite: self.doWrite()

elif control==self.buttonCancel: self.doExit()

def onAction(self, action):

try: F=self.getFocus()

except: F=False

if action == ACTION\_PREVIOUS\_MENU: self.doExit()

elif action == ACTION\_NAV\_BACK: self.doExit()

def CloseWindow(self): self.close()

maxW=1280; maxH=720; W=int(700); H=int(350); L=int((maxW-W)/2); T=int((maxH-H)/2);

TempWindow=MyWindow(L=L,T=T,W=W,H=H,TxtColor=TxtColor,Font=Font,BorderWidth=BorderWidth);

TempWindow.doModal()

del TempWindow

##########################

### Converted to XML

##########################

def contact(msg=""):

class MyWindow(xbmcgui.WindowXMLDialog):

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

self.title = THEME3 % kwargs["title"]

self.image = kwargs["image"]

self.fanart = kwargs["fanart"]

self.msg = THEME2 % kwargs["msg"]

def onInit(self):

self.fanartimage = 101

self.titlebox = 102

self.imagecontrol = 103

self.textbox = 104

self.scrollcontrol = 105

self.showdialog()

def showdialog(self):

self.getControl(self.imagecontrol).setImage(self.image)

self.getControl(self.fanartimage).setImage(self.fanart)

self.getControl(self.fanartimage).setColorDiffuse('9FFFFFFF')

self.getControl(self.textbox).setText(self.msg)

self.getControl(self.titlebox).setLabel(self.title)

self.setFocusId(self.scrollcontrol)

def onAction(self,action):

if action == ACTION\_PREVIOUS\_MENU: self.close()

elif action == ACTION\_NAV\_BACK: self.close()

cw = MyWindow( "Contact.xml" , ADDON.getAddonInfo('path'), 'DefaultSkin', title=ADDONTITLE, fanart=CONTACTFANART, image=CONTACTICON, msg=msg)

cw.doModal()

del cw

def apkInstaller(apk):

class APKInstaller(xbmcgui.WindowXMLDialog):

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

self.shut=kwargs['close\_time']

xbmc.executebuiltin("Skin.Reset(AnimeWindowXMLDialogClose)")

xbmc.executebuiltin("Skin.SetBool(AnimeWindowXMLDialogClose)")

def onClick(self,controlID): self.CloseWindow()

def onAction(self,action):

if action in [ACTION\_PREVIOUS\_MENU, ACTION\_BACKSPACE, ACTION\_NAV\_BACK, ACTION\_SELECT\_ITEM, ACTION\_MOUSE\_LEFT\_CLICK, ACTION\_MOUSE\_LONG\_CLICK]: self.CloseWindow()

def CloseWindow(self):

xbmc.executebuiltin("Skin.Reset(AnimeWindowXMLDialogClose)")

xbmc.sleep(400)

self.close()

xbmc.executebuiltin('Skin.SetString(apkinstaller, Now that %s has been downloaded[CR]Click install on the next window!)' % apk)

popup = APKInstaller('APK.xml', ADDON.getAddonInfo('path'), 'DefaultSkin', close\_time=34)

popup.doModal()

del popup

def speedTest(img):

class speedTest(xbmcgui.WindowXMLDialog):

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

self.imgfile = kwargs['img']

def onInit(self):

self.imagespeed = 101

self.button = 201

self.showdialog()

def showdialog(self):

self.setFocus(self.getControl(self.button))

self.getControl(self.imagespeed).setImage(self.imgfile)

def onClick(self,controlID): self.CloseWindow()

def onAction(self,action):

if action in [ACTION\_PREVIOUS\_MENU, ACTION\_BACKSPACE, ACTION\_NAV\_BACK, ACTION\_SELECT\_ITEM, ACTION\_MOUSE\_LEFT\_CLICK, ACTION\_MOUSE\_LONG\_CLICK]: self.CloseWindow()

def CloseWindow(self):

self.close()

popup = speedTest('SpeedTest.xml', ADDON.getAddonInfo('path'), 'DefaultSkin', img=img)

popup.doModal()

del popup

def Preview(url):

class YTvid(xbmcgui.WindowXMLDialog):

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

self.url = kwargs['url']

def onInit(self):

self.button = 101

self.Obutton = 102

self.showdialog()

def showdialog(self):

self.setFocus(self.getControl(self.Obutton))

if wiz.getCond('System.HasAddon(plugin.video.youtube)') == 1:

url = 'plugin://plugin.video.youtube/play/?video\_id=%s' % self.url

xbmc.Player().play(url, windowed=False)

xbmc.sleep(2000)

if xbmc.Player().isPlayingVideo() == 0:

yt.PlayVideoB(self.url)

def onClick(self,controlID):

if controlId == self.Obutton:

self.close()

else: self.CloseWindow()

def onAction(self,action):

if action in [ACTION\_PREVIOUS\_MENU, ACTION\_BACKSPACE, ACTION\_NAV\_BACK, ACTION\_SELECT\_ITEM, ACTION\_MOUSE\_LEFT\_CLICK, ACTION\_MOUSE\_LONG\_CLICK]: self.CloseWindow(); xbmc.Player().stop()

def CloseWindow(self):

self.close()

YTv = YTvid('Preview.xml', ADDON.getAddonInfo('path'), 'DefaultSkin', url=url)

YTv.doModal()

del YTv

def firstRunSettings():

class firstRun(xbmcgui.WindowXMLDialog):

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

self.whitelistcurrent = kwargs['current']

def onInit(self):

self.title = 101

self.okbutton = 201

self.trakt = 301

self.debrid = 302

self.login = 303

self.alluc = 314

self.profiles = 305

self.advanced = 306

self.favourites = 307

self.superfav = 308

self.repo = 309

self.whitelist = 310

self.cache = 311

self.showdialog()

self.controllist = [self.trakt, self.debrid, self.login,

self.profiles, self.advanced,

self.favourites, self.superfav, self.repo,

self.whitelist, self.cache, self.alluc]

self.controlsettings = ['keeptrakt', 'keepdebrid', 'keeplogin',

'keepprofiles', 'keepadvanced',

'keepfavourites', 'keeprepos', 'keepsuper',

'keepwhitelist', 'clearcache', 'keepalluc']

for item in self.controllist:

if wiz.getS(self.controlsettings[self.controllist.index(item)]) == 'true':

self.getControl(item).setSelected(True)

def showdialog(self):

self.getControl(self.title).setLabel(ADDONTITLE)

self.setFocus(self.getControl(self.okbutton))

def onClick(self, controlId):

if controlId == self.okbutton:

self.close()

for item in self.controllist:

at = self.controllist.index(item)

if self.getControl(item).isSelected(): wiz.setS(self.controlsettings[at], 'true')

else: wiz.setS(self.controlsettings[at], 'false')

if self.getControl(self.whitelist).isSelected() and not self.whitelistcurrent == 'true':

wiz.whiteList('edit')

fr = firstRun( "FirstRunSaveData.xml" , ADDON.getAddonInfo('path'), 'DefaultSkin', current=wiz.getS('keepwhitelist'))

fr.doModal()

del fr

def firstRun():

class MyWindow(xbmcgui.WindowXMLDialog):

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

self.title = THEME3 % ADDONTITLE

self.msg = "Currently no build installed from %s.\n\nSelect 'Build Menu' to install a %s Build or 'Ignore' to never see this message again.\n\nThank you for choosing %s." % (ADDONTITLE,BUILDERNAME, ADDONTITLE)

self.msg = THEME2 % self.msg

def onInit(self):

self.image = 101

self.titlebox = 102

self.textbox = 103

self.buildmenu = 201

self.ignore = 202

self.showdialog()

def showdialog(self):

self.getControl(self.image).setImage(FANART)

self.getControl(self.image).setColorDiffuse('9FFFFFFF')

self.getControl(self.textbox).setText(self.msg)

self.getControl(self.titlebox).setLabel(self.title)

self.setFocusId(self.buildmenu)

def doBuildMenu(self):

wiz.log("[Check Updates] [User Selected: Open Build Menu] [Next Check: %s]" % str(NEXTCHECK), xbmc.LOGNOTICE)

wiz.setS('lastbuildcheck', str(NEXTCHECK))

self.close()

url = 'plugin://%s/?mode=builds' % ADDON\_ID

xbmc.executebuiltin('ActivateWindow(10025, "%s", return)' % url)

def doIgnore(self):

self.close()

wiz.log("[First Run] [User Selected: Ignore Build Menu] [Next Check: %s]" % str(NEXTCHECK), xbmc.LOGNOTICE)

wiz.setS('lastbuildcheck', str(NEXTCHECK))

def onAction(self,action):

if action == ACTION\_PREVIOUS\_MENU: self.doIgnore()

elif action == ACTION\_NAV\_BACK: self.doIgnore()

def onClick(self, controlId):

if (controlId == self.buildmenu): self.doBuildMenu()

else: self.doIgnore()

fr = MyWindow( "FirstRunBuild.xml" , ADDON.getAddonInfo('path'), 'DefaultSkin')

fr.doModal()

del fr

def notification(msg='', test=False):

class MyWindow(xbmcgui.WindowXMLDialog):

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

self.test = kwargs['test']

self.message = THEME2 % kwargs['msg']

def onInit(self):

self.image = 101

self.titlebox = 102

self.titleimage = 103

self.textbox = 104

self.scroller = 105

self.dismiss = 201

self.remindme = 202

self.showdialog()

def showdialog(self):

self.testimage = os.path.join(ART, 'text.png')

self.getControl(self.image).setImage(BACKGROUND)

self.getControl(self.image).setColorDiffuse('9FFFFFFF')

self.getControl(self.textbox).setText(self.message)

self.setFocusId(self.remindme)

if HEADERTYPE == 'Text':

self.getControl(self.titlebox).setLabel(THEME3 % HEADERMESSAGE)

else:

self.getControl(self.titleimage).setImage(HEADERIMAGE)

def doRemindMeLater(self):

if not test == True:

wiz.setS("notedismiss","false")

wiz.log("[Notification] NotifyID %s Remind Me Later" % wiz.getS('noteid'), xbmc.LOGNOTICE)

self.close()

def doDismiss(self):

if not test == True:

wiz.setS("notedismiss","true")

wiz.log("[Notification] NotifyID %s Dismissed" % wiz.getS('noteid'), xbmc.LOGNOTICE)

self.close()

def onAction(self,action):

if action == ACTION\_PREVIOUS\_MENU: self.doRemindMeLater()

elif action == ACTION\_NAV\_BACK: self.doRemindMeLater()

def onClick(self, controlId):

if (controlId == self.dismiss): self.doDismiss()

else: self.doRemindMeLater()

xbmc.executebuiltin('Skin.SetString(headertexttype, %s)' % 'true' if HEADERTYPE == 'Text' else 'false')

xbmc.executebuiltin('Skin.SetString(headerimagetype, %s)' % 'true' if HEADERTYPE == 'Image' else 'false')

notify = MyWindow( "Notifications.xml" , ADDON.getAddonInfo('path'), 'DefaultSkin', msg=msg, test=test)

notify.doModal()

del notify

def updateWindow(name='Testing Window', current='1.0', new='1.1', icon=ICON, fanart=FANART):

class MyWindow(xbmcgui.WindowXMLDialog):

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

self.name = THEME3 % kwargs['name']

self.current = kwargs['current']

self.new = kwargs['new']

self.icon = kwargs['icon']

self.fanart = kwargs['fanart']

self.msgupdate = "Update avaliable for installed build:\n[COLOR %s]%s[/COLOR]\n\nCurrent Version: v[COLOR %s]%s[/COLOR]\nLatest Version: v[COLOR %s]%s[/COLOR]\n\n[COLOR %s]\*Recommened: Fresh install[/COLOR]" % (COLOR1, self.name, COLOR1, self.current, COLOR1, self.new, COLOR1)

self.msgcurrent = "Running latest version of installed build:\n[COLOR %s]%s[/COLOR]\n\nCurrent Version: v[COLOR %s]%s[/COLOR]\nLatest Version: v[COLOR %s]%s[/COLOR]\n\n[COLOR %s]\*Recommended: Fresh install[/COLOR]" % (COLOR1, self.name, COLOR1, self.current, COLOR1, self.new, COLOR1)

def onInit(self):

self.imagefanart = 101

self.header = 102

self.textbox = 103

self.imageicon = 104

self.fresh = 201

self.normal = 202

self.ignore = 203

self.showdialog()

def showdialog(self):

self.getControl(self.header).setLabel(self.name)

self.getControl(self.textbox).setText(THEME2 % self.msgupdate if current < new else self.msgcurrent)

self.getControl(self.imagefanart).setImage(self.fanart)

self.getControl(self.imagefanart).setColorDiffuse('2FFFFFFF')

self.getControl(self.imageicon).setImage(self.icon)

self.setFocusId(self.fresh)

def doFreshInstall(self):

wiz.log("[Check Updates] [Installed Version: %s] [Current Version: %s] [User Selected: Fresh Install build]" % (BUILDVERSION, LATESTVERSION), xbmc.LOGNOTICE)

wiz.log("[Check Updates] [Next Check: %s]" % str(NEXTCHECK), xbmc.LOGNOTICE)

wiz.setS('lastbuildcheck', str(NEXTCHECK))

self.close()

url = 'plugin://%s/?mode=install&name=%s&url=fresh' % (ADDON\_ID, urllib.quote\_plus(BUILDNAME))

xbmc.executebuiltin('RunPlugin(%s)' % url)

def doNormalInstall(self):

wiz.log("[Check Updates] [Installed Version: %s] [Current Version: %s] [User Selected: Normal Install build]" % (BUILDVERSION, LATESTVERSION), xbmc.LOGNOTICE)

wiz.log("[Check Updates] [Next Check: %s]" % str(NEXTCHECK), xbmc.LOGNOTICE)

wiz.setS('lastbuildcheck', str(NEXTCHECK))

self.close()

url = 'plugin://%s/?mode=install&name=%s&url=normal' % (ADDON\_ID, urllib.quote\_plus(BUILDNAME))

xbmc.executebuiltin('RunPlugin(%s)' % url)

def doIgnore(self):

wiz.log("[Check Updates] [Installed Version: %s] [Current Version: %s] [User Selected: Ignore 3 Days]" % (BUILDVERSION, LATESTVERSION), xbmc.LOGNOTICE)

wiz.log("[Check Updates] [Next Check: %s]" % str(THREEDAYS), xbmc.LOGNOTICE)

wiz.setS('lastbuildcheck', str(THREEDAYS))

self.close()

def onAction(self,action):

if action == ACTION\_PREVIOUS\_MENU: self.doIgnore()

elif action == ACTION\_NAV\_BACK: self.doIgnore()

def onClick(self, controlId):

if (controlId == self.fresh): self.doFreshInstall()

elif (controlId == self.normal): self.doNormalInstall()

else: self.doIgnore()

update = MyWindow( "BuildUpdate.xml" , ADDON.getAddonInfo('path'), 'DefaultSkin', name=name, current=current, new=new, icon=icon, fanart=fanart)

update.doModal()

del update