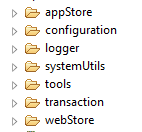
# Project\_2015-5-08 Download Apk from web and upload Davinci

## Requirements Specification

1. Batch download Apk file from Baidu ande Xiaomi web apk store
2. Save the application and game rank info
3. Parse Apk file and get 1.Apk-Name,2.version,3.package-name,4.icon follow the rank
4. Calculate both Apk-File and Apk-ICON MD5 ,then save
5. Upload these information to server api as Json format
6. Upload Apk rank to server api as Jason format
7. Copy Apk file to ftp server an rename as format “MD5.apk” ,also create the floder using first 4 character that from it splite in the middle, like \\127.0.0.1\ab\cd\abcdefg123456789gabcdefg.apk
8. Copy Icon file to ftp server like Apk file.

## Architectural Design Specification

1. Separate main function as 7 module



appStore : Deal with Apk file information and Apk icon file,then compose these info to Json format ,except this appStore module alse move and rename Apk and Icon file into upload buffer floder;

configuration: Parse project config xml file;

logger: This module recorde project running log , logger have five degree info,debug,warn,error and critical, user use like this logger.info(“This is log.”,”on”). Second parameter control first message whether show in console,”on” is show.

systemUtils: This module deal with project framework ,is support module ,content timeout decorate,floder control,module import;

tools: Content three must package file .

transaction: This module is conmmunication section , content multi thread download, upload info or file to server;

webStore: This section deal with different web store ,get web request.

1. Separate web store config from project configuration ,that two different xml. Put web store xml into webStore module as same name with web store PY file. When parse xml ,use ”setattr(self,name,vale) “ this function set each tag name as object attribute name alse set tag value as object attribute value.
2. Create a new app in database

URL: <http://web-service/api/app/register>

JSON:

{

  “version”:”headed-1.0”,

  “heads”:[“md5”, “name”, “package”, “version”, “uri”, “icon\_uri”],

  “apps”: [

    [“12345…edd”,“Angry Bird”,“com.xxx.myfun”,“1.0.1-preview”,“/12/34/12345…edd.apk”,“/ed/bb/edbbx…x.png”],

    [“22345…edd”,“Angry Bird”,“com.xxx.myfun”,“2.0.1-preview”,“/22/34/12345…edd.apk”,“/ed/bb/edbbx…x.png”],

    [“32345…edd”,“Angry Bird”,“com.xxx.myfun”,“3.0.1-preview”,“/32/34/12345…edd.apk”,“/ed/bb/edbbx…x.png”],

         .

         .

         .

    [“N2345…edd”,“Angry Bird”,“com.xxx.myfun”,“n.0.1-preview”,“/N2/34/12345…edd.apk”,“/ed/bb/edbbx…x.png”]

  ]

}

1. Create a new rank in database

URL: <http://web-service/api/app/rank/register>

JSON:

{

  “version” : ”1.0”,

  “store” : ”Baidu/Google/…”,

  “datetime”:”YYYY-MM-DD\_HH:MM:SS”,

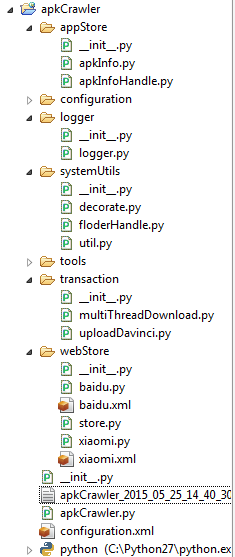
  “category”:”Game/Office/…”,

  “apps”:[“md5-1”, “md5-2”, …, “md5-N”]     // by the order of their ranks

}

1. Use aapt.exe get APK “name”, “package”, “version” information ,and save into a dictionary like {“ Angry Bird”:”AAPT Information….”;”UC browse”:” AAPT Information….”;….}
2. Logger file name like “apkCrawler\_2015\_05\_25\_14\_26\_13.log”,each line star with this format “[LOGGER] 2015-05-25 14:26:20,862 INFO:”.
3. Download APK use multithreading ,accelerate downloading speed.
4. Different web store APK download into different floder ,then rename and move into upload buffer, use xcopy commend copy into server.

## Procedural Design Specification



1. Configuration.xml and baidu.xml

**Configuration.xml**

<default\_aapt\_save\_path>

This config aapt.exe defult path

C:\Users\Administrator\workspace\Davinci\tools\aapt.exe

default\_store

This fill baidu or xiaomi.if more than one store cloud use two same tag.Such as <default\_store>baidu</default\_store>,<default\_store>xiaomi</default\_store>.

<default\_web\_apk\_api>

This is web interface ,receive apk info Json data.

http://web-service/api/app/register

<default\_web\_rank\_api>

This is web interface ,receive apk rank Json data.

http://web-service/api/app/rank/register

<default\_server\_apk\_folder>

This is file server save apk file only.

\\ccr\ec\proj\ssg\dpd\BiTS\DaVinci\_Customers\AppTestCloudApk\Apk

<default\_server\_icon\_folder>

this is file server sasve apk icon file only.

\\ccr\ec\proj\ssg\dpd\BiTS\DaVinci\_Customers\AppTestCloudApk\Icons

**baidu.xml**

<default\_http\_proxy>

Web request need this proxy.

proxy02.cd.intel.com:911

<default\_socks\_proxy>

Web request need this proxy.

proxy.ir.intel.com:1080

<default\_proxy\_type>

Defult “http”

<default\_store\_store>

Defult “baidu”

<default\_apk\_category>

This deside apkcrawler download which list in web store ,defult have tow application and game category tag that means download application and game,if use “application” just download application ,if use “game” just download game.

<default\_download\_begin>

Download apk file from web store index, defult 1

<default\_download\_end>

Download apk file from web store ended, defult 50

<default\_max\_thread\_number>

Multithreading downloading apk file, defult 5

<default\_copy\_file\_method>

This deside upload apk file method ,defult “different” means only upload has update apk file and icon, don’t upload that no change. For decrease traffic use.

<default\_apk\_save\_path>

Local apk save path when crawl from web store.

C:\Users\yangli4x\Downloads\apk\_store\baidu\_store

<default\_apk\_offline\_download\_path>

Local apk save path when crawl from web store.As same like default\_apk\_save\_path.

1. util.py

We import must lib at one util.py and use “from systemUtils.util import \*” in other python file at this project. So user need’t import these lib at each python file ,except one line replace.

1. *# -\*- coding: utf-8 -\*-*
2. import threading
3. import requests
4. import lxml.html
5. import os
6. import sys
7. import shutil
8. import socket
9. import time
10. import md5
11. import json
12. import time
13. import zipfile
14. import logging
15. from logger import logger
16. from systemUtils.decorate import \*
17. import xml.etree.ElementTree as ET
18. from configuration.parseConfig import \*
19. self = ParseConfig()
20. self.load()
21. from appStore.apkInfo import \*
22. from systemUtils.floderHandle import \*
23. from transaction.multiThreadDownload import \*
24. from transaction.uploadDavinci import \*
25. from appStore.apkInfoHandle import \*
26. from webStore.store import \*
27. from webStore.baidu import \*
28. from webStore.xiaomi import \*
29. parseConfig.py

This class parse project config and store config xml. use ”setattr(self,name,vale) “ this function set each tag name as object attribute name alse set tag value as object attribute value. We load project config at first then reload web store config. For adapt different web store use same attribute name .

1. class ParseConfig():
2. def \_\_init\_\_(self):
3. pass
4. def load(self):
5. *# Owner：11602272*
6. *# CreateTime：2015年5月12日*
7. *# ModifyTime：2015年5月12日*
8. *# 函数参数：null*
9. *# 函数方法：解析config.xml*
10. *# 函数返回值：null*
11. xml\_path = os.path.abspath(os.path.join(os.path.split(os.path.realpath(\_\_file\_\_))[0], os.path.pardir))
12. tree = ET.parse(xml\_path + os.path.sep + 'configuration.xml')
13. root = tree.getroot()
14. for child in root:
15. setattr(self, child.tag, child.text)
16. logger.debug("%s = %s" % (child.tag, child.text), "on")
17. store = []
18. for it in root.findall("default\_store"):
19. store.append(it.text)
20. setattr(self, "default\_store", store)
21. logger.debug("default\_store = %s" % store, "on")
22. def reload(self, store):
23. *# Owner：11602272*
24. *# CreateTime：2015年5月19日*
25. *# ModifyTime：*
26. *# 函数参数：null*
27. *# 函数方法：解析webStore文件夹下的web配置文件*
28. *# 函数返回值：null*
29. xml\_path = os.path.abspath(os.path.join(os.path.split(os.path.realpath(\_\_file\_\_))[0], os.path.pardir)) + os.path.sep + "webStore"
30. tree = ET.parse(xml\_path + os.path.sep + store + '.xml')
31. root = tree.getroot()
32. for child in root:
33. setattr(self, child.tag, child.text)
34. logger.debug("%s = %s" % (child.tag, child.text), "on")
35. apkInfo.py

Send aapt dump commend get content, We loop apk list get all apk information and save into dictionary, such as {'cn.jj\_4.02.03.apk': ["package: name='cn.jj' versionCode='40203' , ……. , "densities: '120' '160' '240' '320' '480'\n", "native-code: 'armeabi'\n"];…………..}

So that convenience running when python interoreter get value by key using apk name.

1. class ApkInfo:
2. def \_\_init\_\_(self):
3. pass
4. @timeout(30)
5. def sent\_aapt(self, aapt\_save\_path, apk):
6. *# Owner：11602272*
7. *# CreateTime：2015年5月11日*
8. *# ModifyTime：2015年5月12日*
9. *# 函数参数：传入apk文件*
10. *# 函数方法：发送aapt dump命令*
11. *# 函数返回值：返回aapt反馈信息*
12. command = "%s dump badging %s " % (aapt\_save\_path, apk)
13. aapt\_result = os.popen(command)
14. return aapt\_result
15. @timeout(120)
16. def get\_aapt\_content(self, remote\_apk\_info, apk\_save\_path, aapt\_save\_path):
17. *# Owner：11602272*
18. *# CreateTime：2015年5月11日*
19. *# ModifyTime：2015年5月12日*
20. *# 函数参数：1.文件名，2.apk目录*
21. *# 函数方法：发送aapt dump命令,获取返回的命令*
22. *# 函数返回值：返回aapt反馈信息的数组*
23. aapt\_content\_map = {}
24. if os.path.exists(apk\_save\_path):
25. for i in remote\_apk\_info:
26. file\_name = i[0] + '\_' + i[1] + i[3]
27. if os.path.isfile(apk\_save\_path + os.path.sep + file\_name):
28. aapt\_info = self.sent\_aapt(aapt\_save\_path, apk\_save\_path + os.path.sep + file\_name)
29. aapt\_content\_map[file\_name] = aapt\_info.readlines()
30. logger.info("aapt\_content\_map = %s" % aapt\_content\_map, "on")
31. return aapt\_content\_map
32. logger.py

This class deal with running log , here just show def debug( msg, echo="on"):,

def info( msg, echo="on"):

def warn( msg, echo="on"):

def error( msg, echo="on"):

def critical( msg, echo="on"):

as same like debug(). Each line star with this format “[LOGGER] 2015-05-25 14:26:20,862 INFO:”. User use like this logger.info(“This is log.”,”on”). Second parameter control first message whether show in console, ”on” is show in console.

1. class Logger:
2. """defult Logger"""
3. logger = logging.getLogger("[LOGGER]")
4. file\_handle = logging.FileHandler("apkCrawler\_" + time.strftime("%Y\_%m\_%d\_%H\_%M\_%S", time.localtime()) + ".log")
5. consol\_handle = logging.StreamHandler()
6. formatter = logging.Formatter('%(name)s %(asctime)s %(levelname)s: %(message)s')
7. def debug( msg, echo="on"):
8. logger.setLevel(logging.DEBUG)
9. file\_handle.setLevel(logging.DEBUG)
10. file\_handle.setFormatter(formatter)
11. logger.addHandler(file\_handle)
12. if echo == "on":
13. consol\_handle.setFormatter(formatter)
14. logger.addHandler(consol\_handle)
15. logger.debug(msg)
16. decorate.py

This class is timeout decorate. We use it on top of def like @timeout(60), The program could block in some communication such as get aapt info or download file . So add this decorate function avoid timeout case.

1. class Timeout(Exception):
2. """Function run timeout"""
3. def timeout(seconds):
4. *# Owner：11602272*
5. *# CreateTime：2015年5月15日*
6. *# ModifyTime：*
7. *# 函数参数： 时间*
8. *# 函数方法：若被装饰的方法在指定的时间内未返回，则抛出Timeout异常*
9. *# 函数返回值：function*
10. def timeout\_decorator(func):
11. def \_new\_func(oldfunc, result, oldfunc\_args, oldfunc\_kwargs):
12. result.append(oldfunc(\*oldfunc\_args, \*\*oldfunc\_kwargs))
13. def \_handel\_arg(\*args, \*\*kwargs):
14. result = []
15. new\_kwargs = { *# create new args for \_new\_func, because we want to get the func return val to result list*
16. 'oldfunc': func,
17. 'result': result,
18. 'oldfunc\_args': args,
19. 'oldfunc\_kwargs': kwargs
20. }
21. thd = KThread(target=\_new\_func, args=(), kwargs=new\_kwargs)
22. thd.start()
23. thd.join(seconds)
24. alive = thd.isAlive()
25. thd.kill() *# kill the child thread*
26. if alive:
27. logger.error(u'function %s with args(%s) run too long, timeout %d seconds.' % (func, args, seconds),"on")
28. raise Timeout(u'function %s with args(%s) run too long, timeout %d seconds.' % (func, args, seconds))
29. else:
30. return result[0]
31. return \_handel\_arg
32. return timeout\_decorator
33. multiThreadDownload.py

This class is download module , It’s use multithreading accelrate downloading speed.

User cloud modify max thread number in web store config, defult is 5. def downloader\_multi\_thread have timeout limits in 3600 seconds.

1. WIN\_INVALID\_CHAR\_SET = ['/', '\\', ':', '\*', '?', '"', '<', '>', ',']
2. def \_\_init\_\_(self):
3. pass
4. @timeout(3600)
5. def downloader\_multi\_thread(self, need\_download\_apk\_dictionary, apk\_save\_path, max\_thread\_number, proxy):
6. *# Owner：11602272*
7. *# CreateTime：2015年5月9日*
8. *# ModifyTime：*
9. *# 函数参数：1.需要下载的apk，2.proxy*
10. *# 函数方法：多任务同时并发下载*
11. *# 函数返回值：null*
12. web\_url\_list = need\_download\_apk\_dictionary.values()
13. apk\_name\_list = need\_download\_apk\_dictionary.keys()
14. while web\_url\_list != [] and apk\_name\_list != []:
15. if threading.activeCount() <= int(max\_thread\_number) + 2:
16. t = threading.Thread(target=self.mt\_downloader, args=(apk\_save\_path , apk\_name\_list[0], web\_url\_list[0], proxy))
17. t.start()
18. web\_url\_list.remove(web\_url\_list[0])
19. apk\_name\_list.remove(apk\_name\_list[0])
20. else:
21. time.sleep(3)
22. while threading.activeCount() > 2 :
23. time.sleep(8)
24. logger.info("Please wait, the APP downloading is still underway...", "on")
25. def mt\_downloader(self, apk\_save\_path, apk\_name, web\_url, proxy):
26. *# Owner：11602272*
27. *# CreateTime：2015年5月8日*
28. *# ModifyTime：*
29. *# 函数参数：1.文件名，2.下载地址，3.proxy*
30. *# 函数方法：单个文件下载保存*
31. *# 函数返回值：null*
32. try:
33. logger.info("Downloading start %s " % apk\_name , "on")
34. r = requests.get(web\_url, proxies=proxy)
35. except requests.RequestException, e:
36. pass
37. except socket.error, e:
38. pass
39. else:
40. try:
41. if os.path.exists(apk\_save\_path) == False:
42. FloderHandel().create\_apk\_folder(apk\_save\_path)
43. with open (apk\_save\_path + os.path.sep + apk\_name, 'wb') as apk:
44. apk.write(r.content)
45. except IOError as e:
46. logger.error("Failed to write %s due to IO error" % apk\_name, "on")
47. except:
48. pass
49. if os.path.isfile(apk\_save\_path + os.path.sep + apk\_name):
50. logger.info("Downloaded Finish %s" % apk\_name , "on")
51. store.py

This class is superclass, baidu.py and xiaomi.py inherit this class. def get\_url\_content have timeout limits in 30 seconds. def get\_apk\_list have timeout limits in 60 seconds.

1. class Store:
2. def \_\_init\_\_(self):
3. pass
4. @timeout(30)
5. def get\_url\_content(self, web\_url, proxy=None):
6. *# Owner：11602272*
7. *# CreateTime：2015年5月15日*
8. *# ModifyTime：*
9. *# 函数参数： 请求地址，Proxy*
10. *# 函数方法：访问被请求的web\_url返回响应结果*
11. *# 函数返回值：req*
12. try:
13. req = requests.get(web\_url, proxies=proxy)
14. except requests.exceptions.ConnectionError as e:
15. logger.error('Connection was lost,please check the network.', "on")
16. sys.exit(1)
17. except Exception, e:
18. logger.error("error is %s" % e, "on")
19. sys.exit(1)
20. else:
21. return req
22. @timeout(60)
23. def get\_apk\_list(self):
24. *# Owner：11602272*
25. *# CreateTime：2015年5月15日*
26. *# ModifyTime：*
27. *# 函数参数：*
28. *# 函数方法：返回指定排名的app列表，列表包括APP名字，版本号，下载链接*
29. *# 函数返回值：remote\_apk\_info,['com.tianmashikong.qmqj.bd', '1.4.1', 'http://gdown.baidu.com/data/wisegame/cce33f84449b99f2/quanminqiji\_141.apk', '.apk']*
30. remote\_apk\_info = []
31. return remote\_apk\_info
32. baidu.py、xiaomi.py

This python class is get apk rank list and download url at web store.

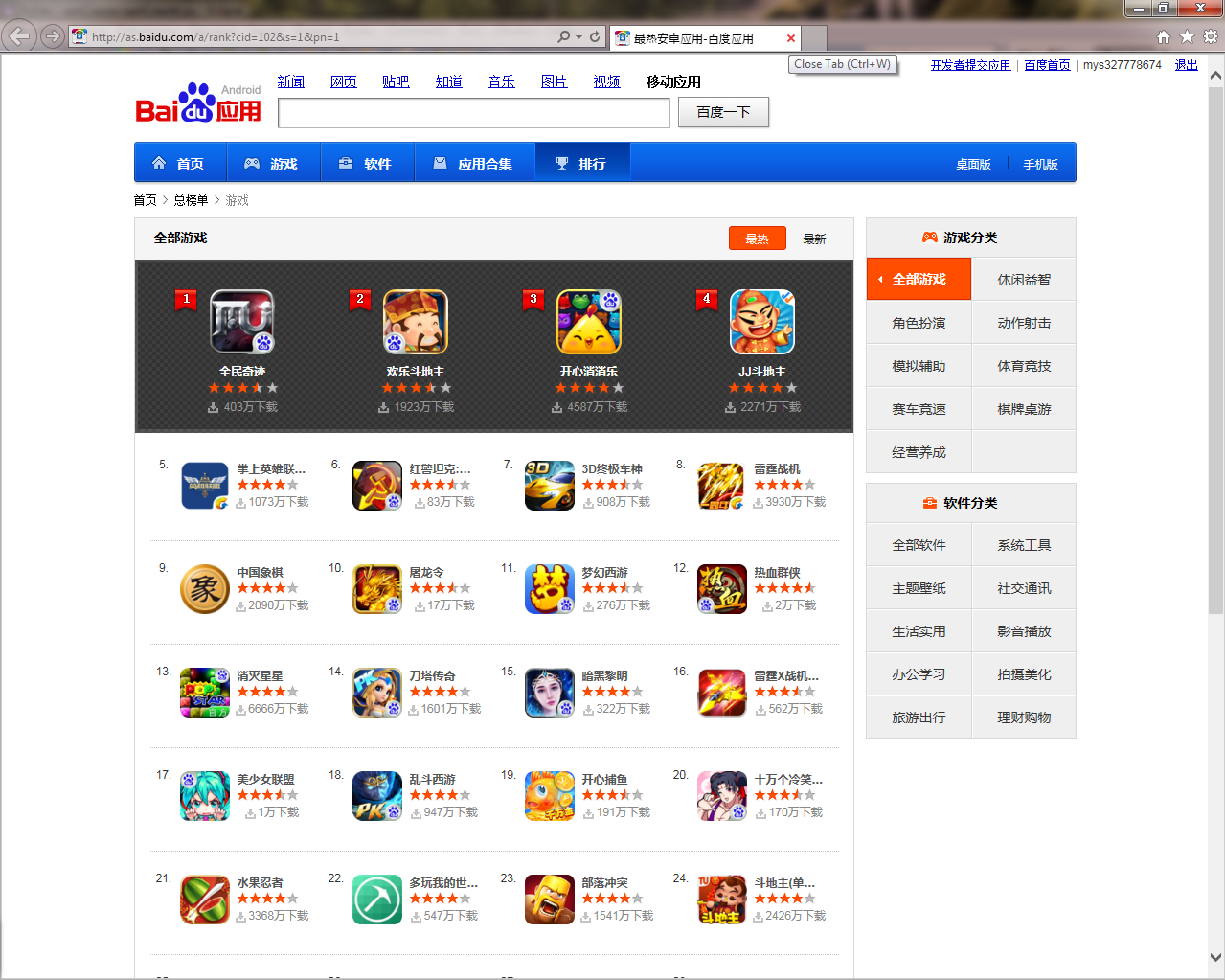
1. class Baidu(Store):
2. GENERAL\_FILE\_EXTENSION = ".apk"
3. def \_\_init\_\_(self):
4. pass
5. @timeout(30)
6. def get\_url\_content(self, web\_url, proxy=None):
7. *# Owner：11602272*
8. *# CreateTime：2015年5月15日*
9. *# ModifyTime：*
10. *# 函数参数： 请求地址，Proxy*
11. *# 函数方法：访问被请求的web\_url返回响应结果*
12. *# 函数返回值：req*
13. return Store.get\_url\_content(self, web\_url, proxy)
14. @timeout(60)
15. def get\_apk\_list(self, proxy, download\_begin, download\_end, apk\_category):
16. ……………
17. uploadDavinci.py

This class could upload Apk info and rank to web server api, and cloud copy file that in upload buffer floder to file server then clean upload buffer floder.

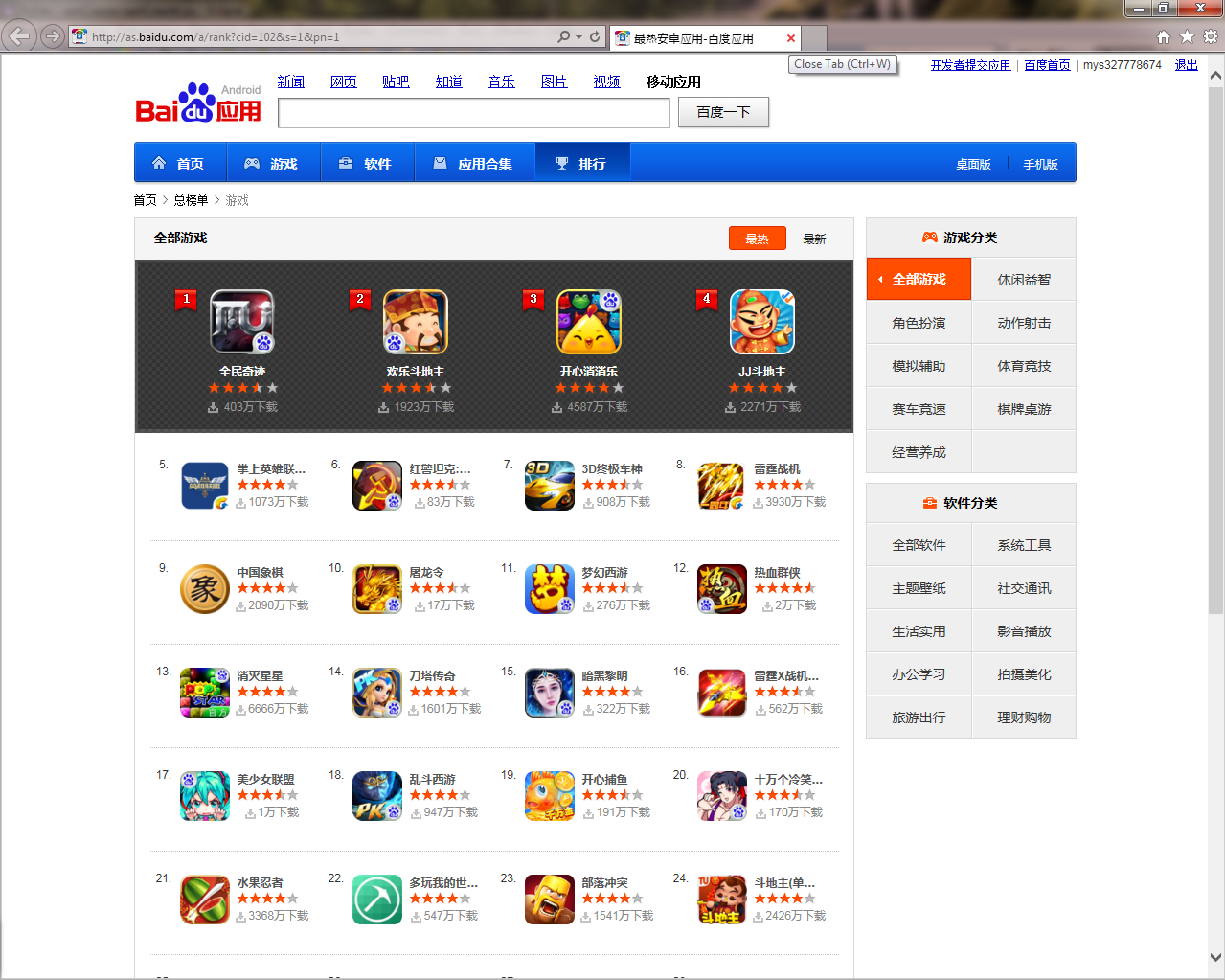
1. class Upload:
2. def \_\_init\_\_(self):
3. pass
4. @timeout(20)
5. def send\_to\_web\_server(self, web\_api, data):
6. *# Owner：11602272*
7. *# CreateTime：2015年5月11日*
8. *# ModifyTime：*
9. *# 函数参数：1.服务器接收地址，2.Jason格式的数据*
10. *# 函数方法：上传数据到服务器接口*
11. *# 函数返回值：null*
12. for upload in data:
13. r = requests.post(web\_api, upload)
14. logger.debug("Web response is %s" % r.content, "on")
15. logger.info('Successfully! Please see apkCrawler log.', "on")
16. def copy\_to\_file\_server(self, apk\_save\_path, server\_apk\_folder, server\_icon\_folder):
17. *# Owner：11602272*
18. *# CreateTime：2015年5月20日*
19. *# ModifyTime：*
20. *# 函数参数：1.本地apk保存跟目录，2服务器保存apk的根目录，3.服务器保存icon的根目录*
21. *# 函数方法：上传apk和icon到文件服务器并清空本地文件夹*
22. *# 函数返回值：null*
23. local\_apk\_upload\_path = apk\_save\_path + os.path.sep + "upload" + os.path.sep + "apk"
24. local\_icon\_upload\_path = apk\_save\_path + os.path.sep + "upload" + os.path.sep + "icon"
25. if os.path.isdir(local\_apk\_upload\_path):
26. commend = "xcopy /e /s /y %s\\* %s" % (local\_apk\_upload\_path, server\_apk\_folder)
27. re = os.system(commend)
28. logger.debug("commend is %s return %s " % (commend, re), "on")
29. if re == 0 :
30. if os.path.isdir(local\_apk\_upload\_path):
31. shutil.rmtree(local\_apk\_upload\_path)
32. if os.path.isdir(local\_icon\_upload\_path):
33. commend = "xcopy /e /s /y %s\\* %s" % (local\_icon\_upload\_path, server\_icon\_folder)
34. re = os.system(commend)
35. logger.debug("commend is %s return %s " % (commend, re), "on")
36. if re == 0 :
37. if os.path.isdir(local\_icon\_upload\_path):
38. shutil.rmtree(local\_icon\_upload\_path)

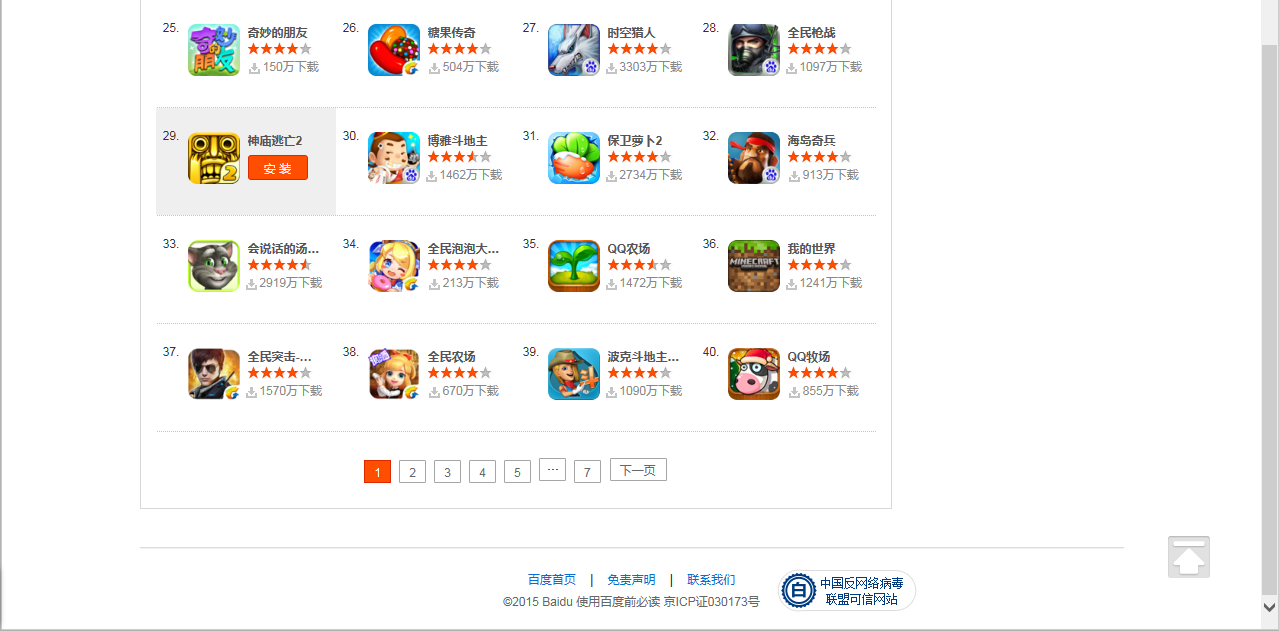
## Test Plan & Report

Baidu application web store : <http://as.baidu.com/a/rank?cid=101&s=1&pn=1>



Baidu game web store : <http://as.baidu.com/a/rank?cid=102&s=1&pn=1>



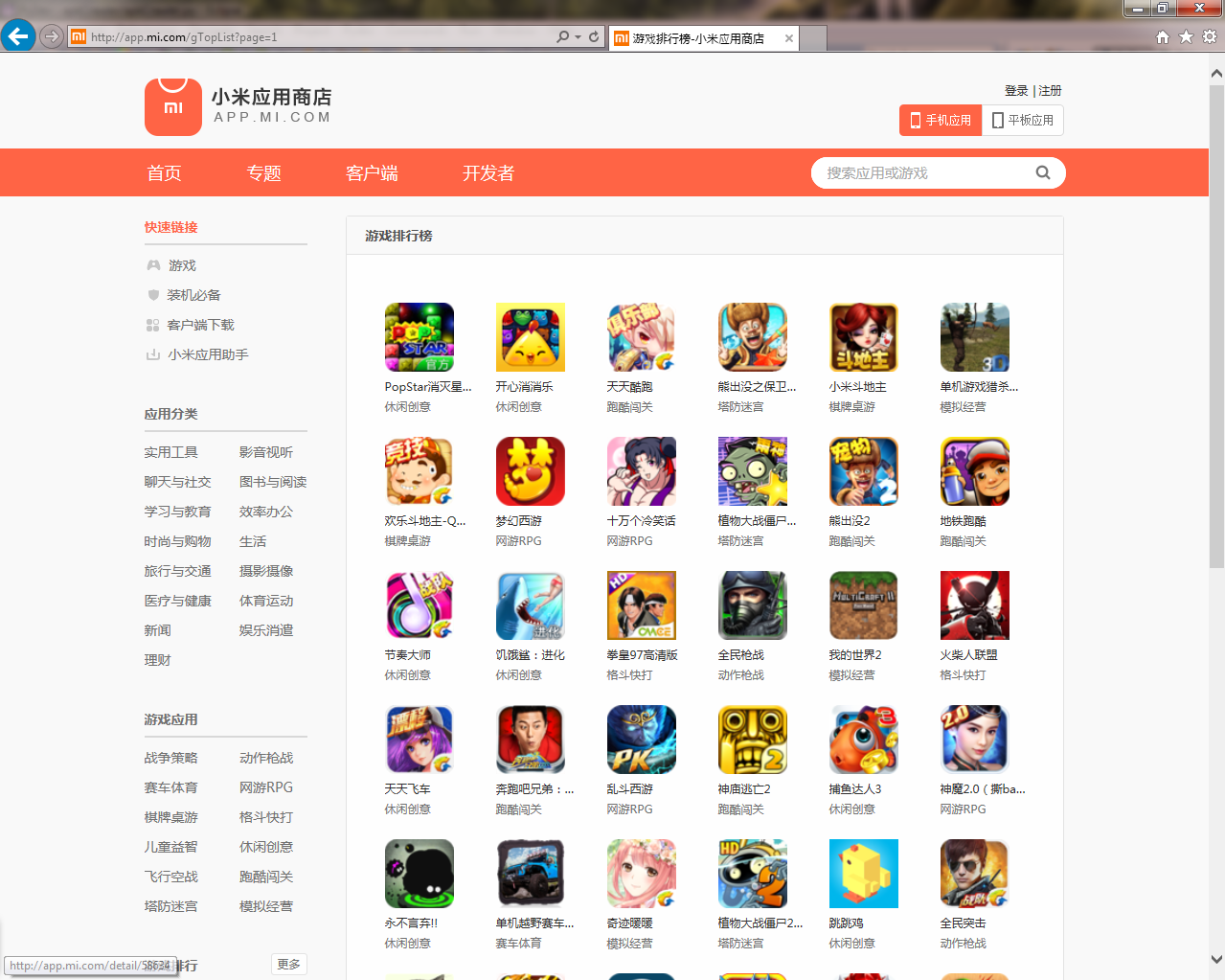


At Baidu web store Application and Game category only have 7 page, 40 item in one page.

Xiaomi application web store : <http://app.mi.com/topList?page=1>



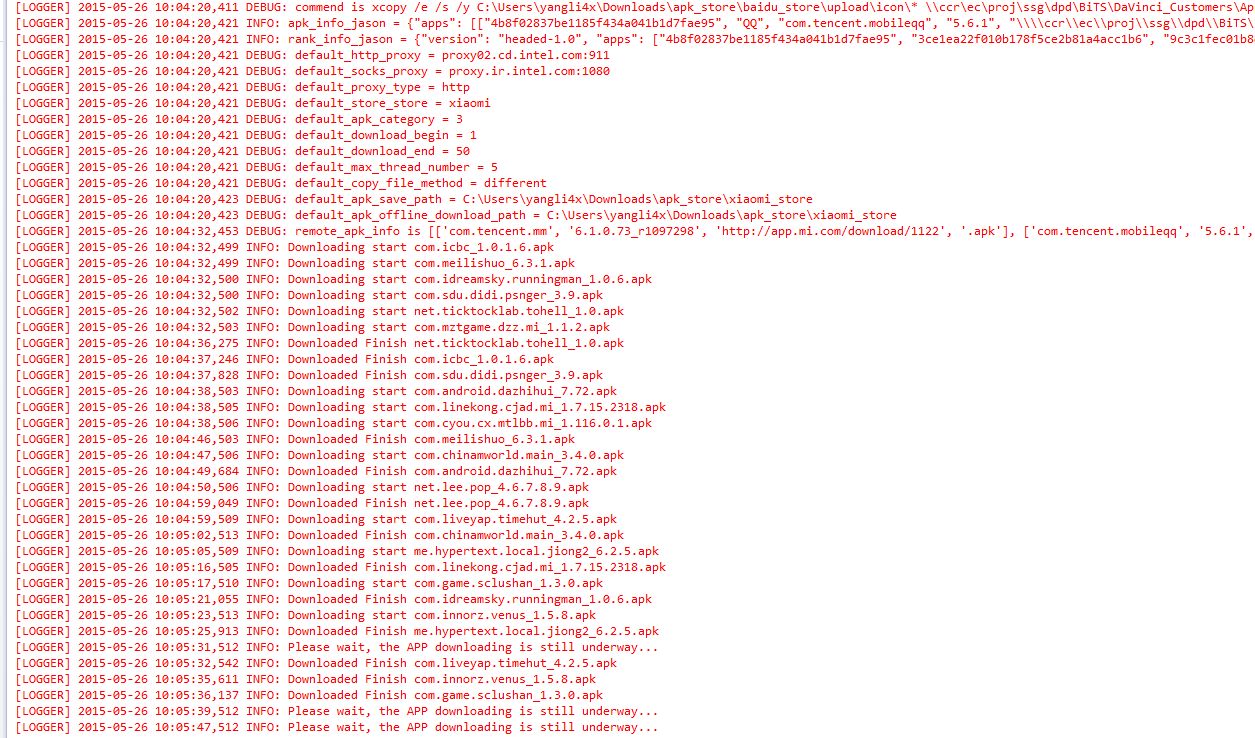
Xiaomi game web store : <http://app.mi.com/gTopList?page=1>



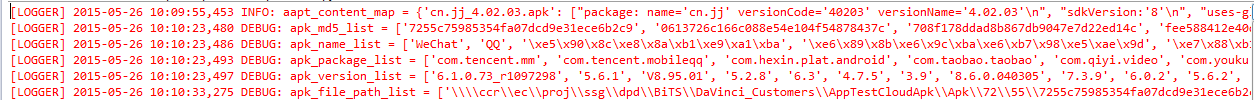
Running log : 

Crawl Xiaomi web store,using defult config running log Such as:

Prase local project config and web store config then get web request and multithreading download Apk file:



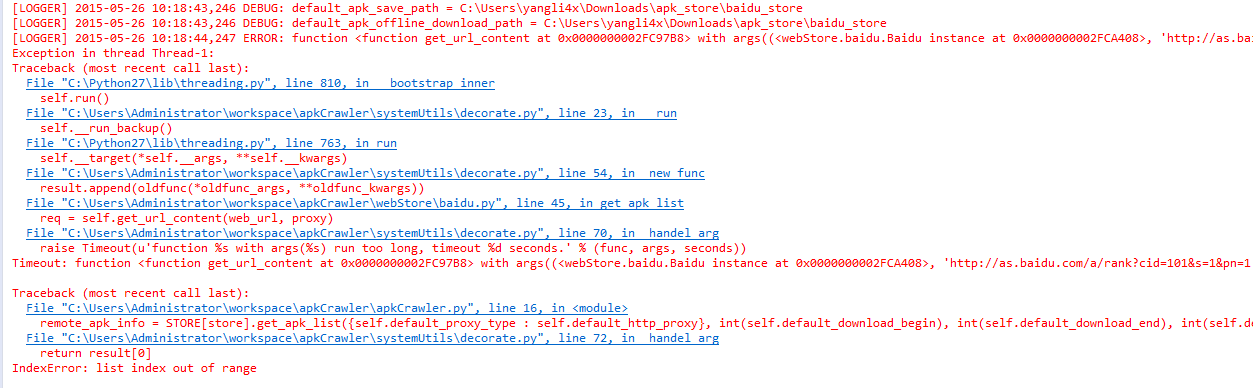
Get Information from Apk aapt dump :



Upload Apk file and Apk Icon to server: 

Send Apk info and rank to web api:

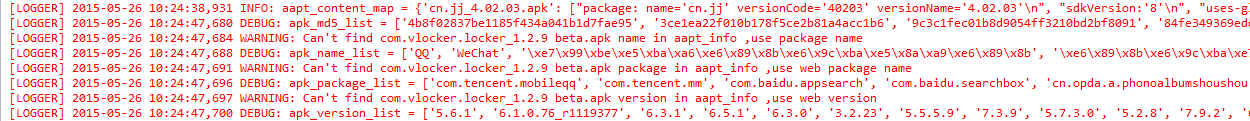


Timeout exception log such as: 

Timeout: function <function get\_url\_content at 0x0000000002FC97B8> with args((<webStore.baidu.Baidu instance at 0x0000000002FCA408>, 'http://as.baidu.com/a/rank?cid=101&s=1&pn=1', {'http': 'proxy02.cd.intel.com:911'})) run too long, timeout 1 seconds.

This log show timeout limit too short, 1 seconds can’t finished.

Warning Log such as :



WARNING: Can't find com.vlocker.locker\_1.2.9 beta.apk package in aapt\_info ,use web package name

This log show can’t found Apk package name in Apk aapt dump content.

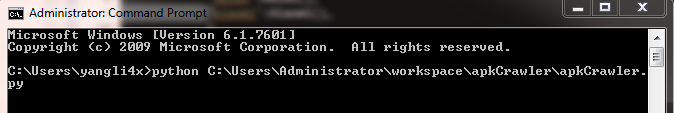
## User Guide

Project config is configuration.xml at root path,and store config at webStore floder.

User need or not modify default\_apk\_category,default\_download\_begin,default\_download\_end, default\_max\_thread\_number…decide download whitch category from web stroe rank index and how much need ,also decide multithreading download number.

Other config tag item and value already explain in C. Procedural Design Specification a)Configuration.xml and baidu.xml.

Project entry “apkCrawler.py” with no parameter. In commend line “python C:\Users\Administrator\workspace\apkCrawler\apkCrawler.py”, like



# Project\_2015-6-3 Expand baidu and xiaomi web store Sub-category

## Requirements Specification

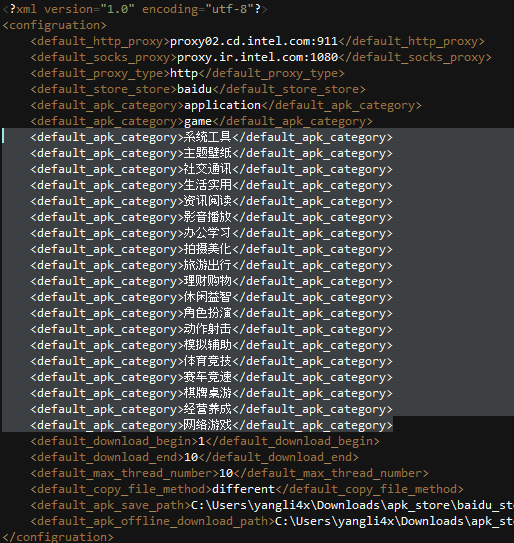
1. Add all sub-category on baidu web store into ApkCrawler, and upload rank into web api.
2. Add all sub-category on xiaomi web store into ApkCrawler, and upload rank into web api.

## Architectural Design Specification

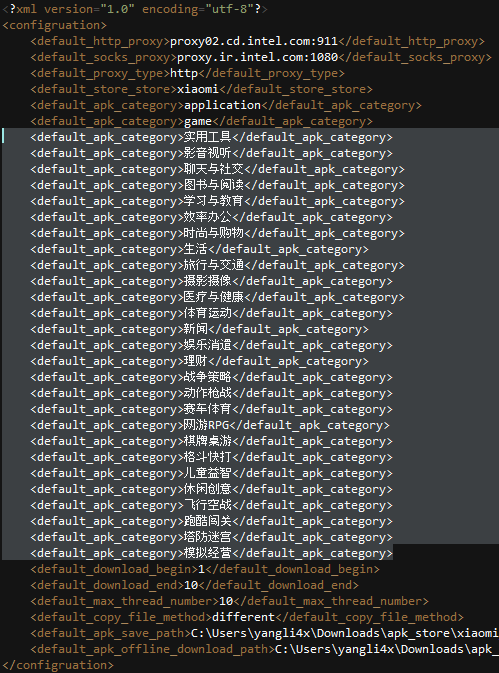
1. Append all sub-category into baidu.xml and xiaomi.xml differently
2. Modify baidu.py and xiaomi.py achieve each sub-category.
3. Using a map matching sub-category with web url
4. Iterate the map make up each whole web url
5. Upload ranking list.

## Procedural Design Specification

1. baidu.xml



1. xiaomi.xml



1. baidu.py

Add a dictionary matching sub-category with web url info, return category\_selection

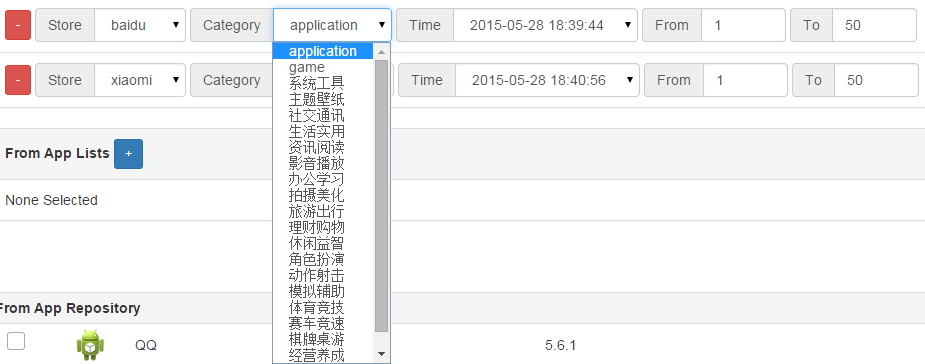
1. def get\_category\_info(self, apk\_category):
2. *# Owner：11602272*
3. *# CreateTime：2015年6月1日*
4. *# ModifyTime：*
5. *# 函数参数： 用户配置的下载标签*
6. *# 函数方法：匹配web标签的web\_uil信息*
7. *# 函数返回值：category\_selection*
8. category\_selection = []
9. url = {"application":"101", "game":"102",
10. u"系统工具":["software", "501"], u"主题壁纸":["software", "502"],
11. u"社交通讯":["software", "503"], u"生活实用":["software", "504"],
12. u"资讯阅读":["software", "505"], u"影音播放":["software", "506"],
13. u"办公学习":["software", "507"], u"拍摄美化":["software", "508"],
14. u"旅游出行":["software", "509"], u"理财购物":["software", "510"],
15. u"休闲益智":["game", "401"], u"角色扮演":["game", "402"],
16. u"动作射击":["game", "403"], u"模拟辅助":["game", "404"],
17. u"体育竞技":["game", "405"], u"赛车竞速":["game", "406"],
18. u"棋牌桌游":["game", "407"], u"经营养成":["game", "408"],
19. u"网络游戏":["game", ""],
20. }
21. for category in apk\_category:
22. category\_selection.append(url.get(category))
23. return category\_selection
24. @timeout(60)
25. def get\_apk\_list(self, proxy, download\_begin, download\_end, apk\_category):
26. *# Owner：11602272*
27. *# CreateTime：2015年5月8日*
28. *# ModifyTime：2015年6月2日*
29. *# 函数参数：1.Proxy,2.排名开始，3.排名结束，4.APP分类类型*
30. *# 函数方法：返回指定排名的app列表，列表包括APP名字，版本号，下载链接*
31. *# 函数返回值：all\_download\_info,['com.tianmashikong.qmqj.bd', '1.4.1', 'http://gdown.baidu.com/data/wisegame/cce33f84449b99f2/quanminqiji\_141.apk', '.apk']*
32. *# ("",1,600,2,r"C:\Users\yangli4x\Downloads\Baidu",r"C:\Users\yangli4x\Downloads\Baidu600")*
33. remote\_apk\_info = []
34. category\_selection = self.get\_category\_info(apk\_category)
35. *# logger.debug("category\_selection is %s" % category\_selection, "on")*
36. apk\_number\_per\_page = 40
37. for cid in category\_selection:
38. page\_number = download\_end / apk\_number\_per\_page + 1
39. if download\_end % apk\_number\_per\_page != 0:
40. page\_number = page\_number + 1
41. count = 0
42. if cid == "101" or cid == "102":
43. for i in xrange(1, page\_number):
44. web\_url = "http://as.baidu.com/a/rank?cid=" + cid + "&s=1&pn=" + str(i)
45. *# http://as.baidu.com/a/rank?cid=0&s=101&pn=1*
46. req = self.get\_url\_content(web\_url, proxy)
47. content = req.content.decode("utf-8")
48. tree = lxml.html.fromstring(content)
49. all\_match = tree.xpath("//a[@data-download\_url]")
50. for i in range(0, len(all\_match)):
51. if count < download\_begin - 1:
52. count = count + 1
53. continue
54. count = count + 1
55. if count > download\_end:
56. break
57. remote\_apk\_info.append([all\_match[i].values()[7], all\_match[i].values()[5], all\_match[i].values()[8], self.GENERAL\_FILE\_EXTENSION])
58. else:
59. web\_url = "http://shouji.baidu.com/" + cid[0] + "/list?cid=" + str(cid[1])
60. *# logger.debug("web\_url is %s" % web\_url, "on")*
61. *# http://shouji.baidu.com/software/list?cid=501*
62. req = self.get\_url\_content(web\_url, proxy)
63. content = req.content.decode("utf-8")
64. tree = lxml.html.fromstring(content)
65. all\_match = tree.xpath("//span[@data\_url]")
66. for i in range(0, len(all\_match)):
67. if count < download\_begin - 1:
68. count = count + 1
69. continue
70. count = count + 1
71. if count > download\_end:
72. break
73. remote\_apk\_info.append([all\_match[i].values()[10], all\_match[i].values()[11], all\_match[i].values()[7], self.GENERAL\_FILE\_EXTENSION])
74. logger.debug("remote\_apk\_info is %s" % remote\_apk\_info, "on")
75. return remote\_apk\_info
76. Xiaomi.py

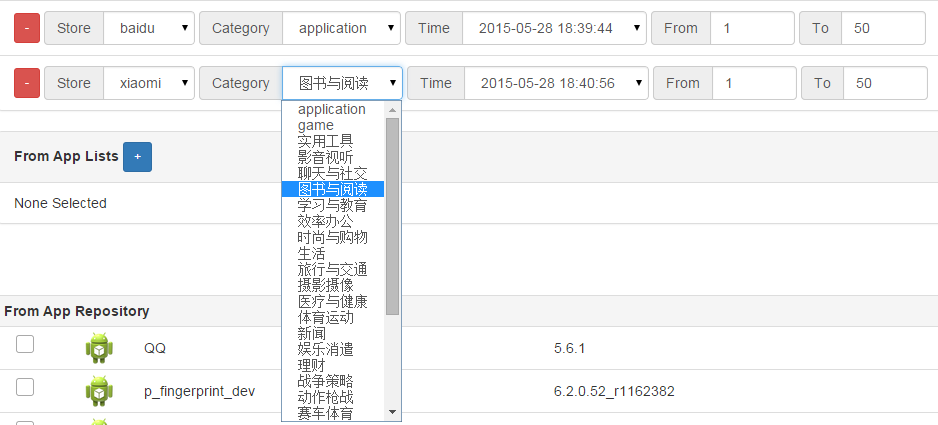
Add a dictionary matching sub-category with web url info, return category\_selection

1. def get\_category\_info(self, apk\_category):
2. *# Owner：11602272*
3. *# CreateTime：2015年6月1日*
4. *# ModifyTime：*
5. *# 函数参数： 用户配置的下载标签*
6. *# 函数方法：匹配web标签的web\_uil信息*
7. *# 函数返回值：category\_selection*
8. category\_selection = []
9. url = {"application":"topList", "game":"gTopList",
10. u"实用工具":"5", u"影音视听":"27", u"聊天与社交":"2", u"图书与阅读":"7",
11. u"学习与教育":"12", u"效率办公":"10", u"时尚与购物":"9", u"生活":"4",
12. u"旅行与交通":"3", u"摄影摄像":"6", u"医疗与健康":"14", u"体育运动":"8",
13. u"新闻":"11", u"娱乐消遣":"13", u"理财":"1",
14. u"战争策略":"16", u"动作枪战":"17", u"赛车体育":"18", u"网游RPG":"19",
15. u"棋牌桌游":"20", u"格斗快打":"21", u"儿童益智":"22", u"休闲创意":"23",
16. u"飞行空战":"25", u"跑酷闯关":"26", u"塔防迷宫":"28", u"模拟经营":"29",
17. }
18. for category in apk\_category:
19. *# logger.debug("category is %s , value is %s" % (category, url.get(category)), "on")*
20. category\_selection.append(url.get(category))
21. return category\_selection
22. @timeout(120)
23. def get\_apk\_list(self, proxy, download\_begin, download\_end, apk\_category):
24. *# Owner：11602272*
25. *# CreateTime：2015年5月8日*
26. *# ModifyTime：2015年6月2日*
27. *# 函数参数：1.Proxy,2.排名开始，3.排名结束，4.APP分类类型*
28. *# 函数方法：返回指定排名的app列表，列表包括APP名字，版本号，下载链接*
29. *# 函数返回值：all\_download\_info,['com.tianmashikong.qmqj.bd', '1.4.1', 'http://gdown.baidu.com/data/wisegame/cce33f84449b99f2/quanminqiji\_141.apk', '.apk']*
30. *# ("",1,600,2,r"C:\Users\yangli4x\Downloads\Baidu",r"C:\Users\yangli4x\Downloads\Baidu600")*
31. remote\_apk\_info = []
32. category\_selection = self.get\_category\_info(apk\_category)
33. logger.debug("category\_selection is %s" % category\_selection, "on")
34. xiaomi\_store\_prefix = "http://app.mi.com/"
35. for cid in category\_selection:
36. if cid == "topList" or cid == "gTopList":
37. apk\_number\_per\_page = 48
38. page\_number = download\_end / apk\_number\_per\_page
39. if download\_end < apk\_number\_per\_page or download\_end % apk\_number\_per\_page != 0:
40. page\_number = page\_number + 1
41. count = 0
42. for i in range(1, page\_number + 1):
43. url = xiaomi\_store\_prefix + cid + '?page=' + str(i)
44. req = self.get\_url\_content(url, proxy)
45. content = req.content.decode("utf-8")
46. tree = lxml.html.fromstring(content)
47. all\_match = tree.xpath("//h5//a[@href]")
48. for e in all\_match:
49. if e.values() != '':
50. if count < download\_begin - 1:
51. count = count + 1
52. continue
53. count = count + 1
54. if count > download\_end:
55. break
56. detail\_url = xiaomi\_store\_prefix[:-1] + e.values()[0]
57. req = requests.get(detail\_url, proxies=proxy)
58. content = req.content.decode("utf-8")
59. tree = lxml.html.fromstring(content)
60. dl\_url\_match = tree.xpath("//a[@class='download']")
61. detail\_info\_match = tree.xpath("//ul[@class=' cf']//li")
62. *# logger.debug("detail\_info\_match is %s" % detail\_info\_match, "on")*
63. *# if we want to save apk name in Chinese, please set the apk\_name eqaul i.text*
64. *# apk\_name = i.text*
65. apk\_name = detail\_info\_match[7].text
66. apk\_version = detail\_info\_match[3].text
67. apk\_dl\_url = xiaomi\_store\_prefix[:-1] + dl\_url\_match[0].values()[0]
68. remote\_apk\_info.append([apk\_name, apk\_version, apk\_dl\_url, self.GENERAL\_FILE\_EXTENSION])
69. else:
70. apk\_number\_per\_page = 30
71. page\_number = (download\_end - 12) / apk\_number\_per\_page
72. if (download\_end - 12) < apk\_number\_per\_page or (download\_end - 12) % apk\_number\_per\_page != 0:
73. page\_number = page\_number + 1
74. count = 0
75. *# logger.debug("cid is %s ,page\_number is %s" % (cid, page\_number), "on")*
76. for i in range(0, page\_number + 1):
77. url = xiaomi\_store\_prefix + 'category/' + str(cid) + '#page=' + str(i)
78. *# logger.debug("url is %s" % url, "on")*
79. req = self.get\_url\_content(url, proxy)
80. content = req.content.decode("utf-8")
81. tree = lxml.html.fromstring(content)
82. all\_match = tree.xpath("//h5//a[@href]")
83. if i > 0:
84. all\_match = all\_match[12:]
85. for e in all\_match:
86. if e.values() != '':
87. if count < download\_begin - 1:
88. count = count + 1
89. continue
90. count = count + 1
91. if count > download\_end:
92. break
93. detail\_url = xiaomi\_store\_prefix[:-1] + e.values()[0]
94. req = requests.get(detail\_url, proxies=proxy)
95. content = req.content.decode("utf-8")
96. tree = lxml.html.fromstring(content)
97. dl\_url\_match = tree.xpath("//a[@class='download']")
98. detail\_info\_match = tree.xpath("//ul[@class=' cf']//li")
99. *# logger.debug("detail\_info\_match is %s" % detail\_info\_match, "on")*
100. *# if we want to save apk name in Chinese, please set the apk\_name eqaul i.text*
101. *# apk\_name = i.text*
102. apk\_name = detail\_info\_match[7].text
103. apk\_version = detail\_info\_match[3].text
104. apk\_dl\_url = xiaomi\_store\_prefix[:-1] + dl\_url\_match[0].values()[0]
105. remote\_apk\_info.append([apk\_name, apk\_version, apk\_dl\_url, self.GENERAL\_FILE\_EXTENSION])
106. else:
107. pass
108. logger.debug("remote\_apk\_info is %s" % remote\_apk\_info, "on")
109. return remote\_apk\_info

## Test Plan & Report

1. Baidu sub-category



1. Xiaomi sub-category

## User Guide

1. When user crawl apk need config xml choose whitch category need to be download.

# Project\_2015-7-20 Independent script for client to download apk from our server

## Requirements Specification

1. Need user could download apk from internal server.
2. The script could self-update when it’s updated.
3. Create a rank list into user directory.
4. The rank list record the Apk what missed reason.
5. Let user select Store, category, rank range, rank history.

## Architectural Design Specification

1. Create class “Config”, record some default parmeter.
2. Create class “WatchDog”, Keep listening the script , When find script different with server, The Watchdog will update script and reload it-self.
3. Create class “DownloadByRank”, read server each store Json file.
4. Use different timer in Win and Linux, limit user input, when user choose timeout will use default value.
5. Use “retry” decoration, let user could reselect when input error.

## Procedural Design Specification

1. Class config.

class Config:

default\_apk\_local\_path = r"C:\Users\%s\Downloads\apk-downloader" % getpass.getuser()

default\_server\_apk\_folder = r"\\ccr\ec\proj\ssg\dpd\BiTS\DaVinci\_Customers\AppTestCloudApk\Apk"

default\_rank\_min = 1

default\_rank\_max = 0

default\_plantform = ["arm\_Phone", "arm\_Tablet", "x86\_Phone", "x86\_Tablet"]

default\_build\_time = "2015-8-04\_13-24-56"

1. Class WatchDog.

class WatchDog:

*# Owner：11602272*

*# CreateTime：2015年7月14日*

*# ModifyTime：2015年7月14日*

*# 类参数： null*

*# 函数方法：一旦Python脚本发生改动，自动重新启动被监听的方法*

def iter\_module\_files(self):

for module in sys.modules.values():

file\_name = getattr(module, '\_\_file\_\_', None)

if file\_name:

if file\_name[-4:] in ('.pyo', '.pyc'):

file\_name = file\_name[:-1]

yield file\_name

def is\_any\_file\_changed(self, mtimes):

for file\_name in self.iter\_module\_files():

try:

mtime = os.stat(file\_name).st\_mtime

print "mtime", mtime

except IOError:

continue

old\_time = mtimes.get(file\_name, None)

if old\_time is None:

mtimes[file\_name] = mtime

elif mtime > old\_time:

return 1

return 0

def start\_change\_detector(self):

mtimes = {}

while 1:

if self.is\_any\_file\_changed(mtimes):

sys.exit(3)

time.sleep(1)

def restart\_with\_reloader(self):

while 1:

args = [sys.executable] + sys.argv

new\_env = os.environ.copy()

new\_env['RUN\_FLAG'] = 'true'

exit\_code = subprocess.call(args, env=new\_env)

if exit\_code != 3:

return exit\_code

def run\_with\_reloader(self, fun):

if os.environ.get('RUN\_FLAG') == 'true':

thread.start\_new\_thread(fun, ())

try:

self.start\_change\_detector()

except KeyboardInterrupt:

pass

else:

try:

sys.exit(self.restart\_with\_reloader())

except KeyboardInterrupt:

pass

1. Select platform

def assert\_platform(self):

*# print "platform.system()",platform.system()*

if 'Windows' in platform.system():

return "Windows"

elif 'Linux' in platform.system() or "CYGWIN" in platform.system():

return "Linux"

1. Achieve user input

def get\_user\_input(self, caption, default, hidden\_flag=False, timeout=10):

*# Owner：11602272*

*# CreateTime：2015年7月3日*

*# ModifyTime：*

*# 函数参数： 说明标题, 默认值*

*# 函数方法：获取用户输入的值*

*# 函数返回值：user\_input*

sys.stdout.write('%s(Default is %s):\n' % (caption, default));

os\_platform = self.assert\_platform()

if os\_platform == "Windows":

import msvcrt

start\_time = time.time()

user\_input = ''

while True:

if msvcrt.kbhit():

if hidden\_flag:

user\_input\_char = msvcrt.getch()

else:

user\_input\_char = msvcrt.getche()

if ord(user\_input\_char) == 13: *# enter\_key*

break

elif ord(user\_input\_char) >= 32: *# space\_char*

user\_input += user\_input\_char

if hidden\_flag:

sys.stdout.write('\*')

if len(user\_input) == 0 and (time.time() - start\_time) > timeout:

break

elif os\_platform == "Linux":

import select

print "You have ten seconds to answer!"

i, o, e = select.select([sys.stdin], [], [], timeout)

if (i):

user\_input = sys.stdin.readline().strip()

else:

user\_input = ""

if len(user\_input) > 0:

return user\_input

else:

return default

1. Write rank.csv

with open (apk\_offline\_download\_path + os.path.sep + "rank.csv", "a") as rank:

for i in dict["apps"]:

rank.write(i["name"] + ", " + i["package"] + ", " + i["version"] + ", " + i["apk\_path"] + ", " + str(i["debug\_info"]) + "\n")

1. Chinese output

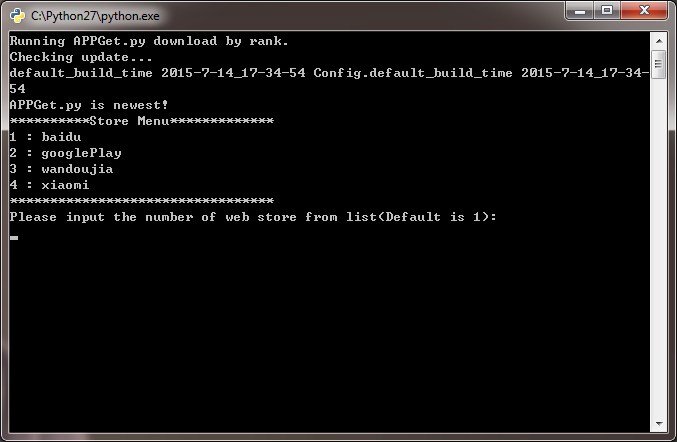
import sys

reload(sys)

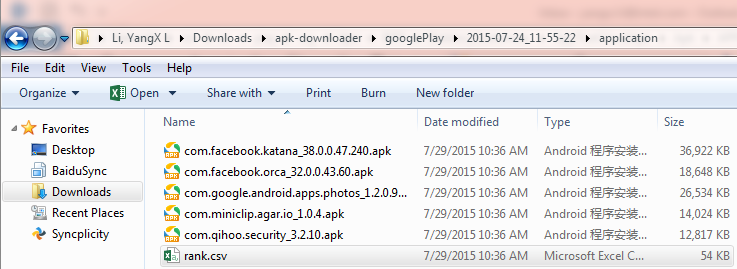
sys.setdefaultencoding("utf-8")

## Test Plan & Report

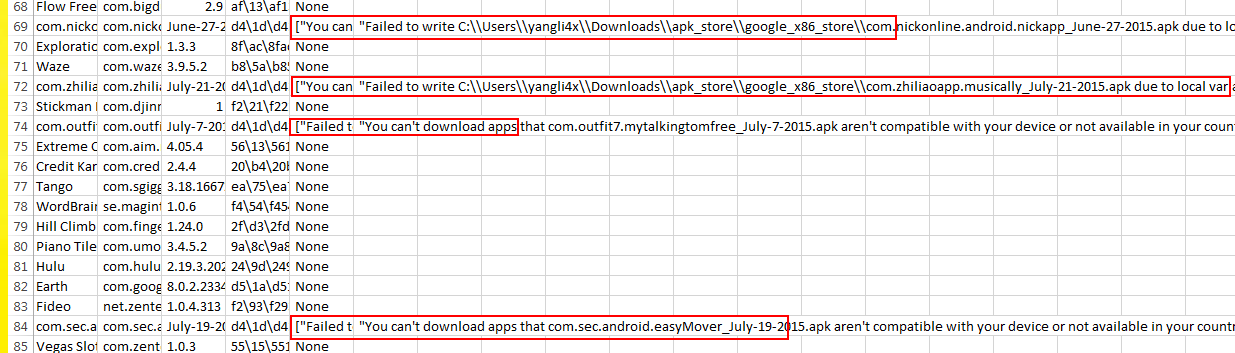
1. User run interface



1. Downloaded



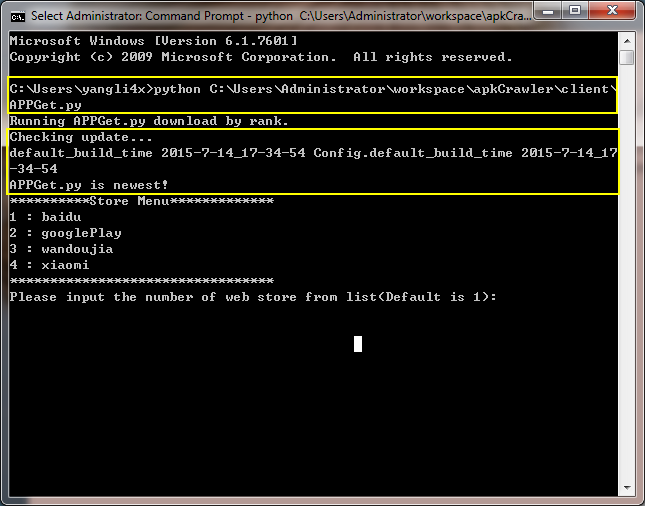
1. Achieved



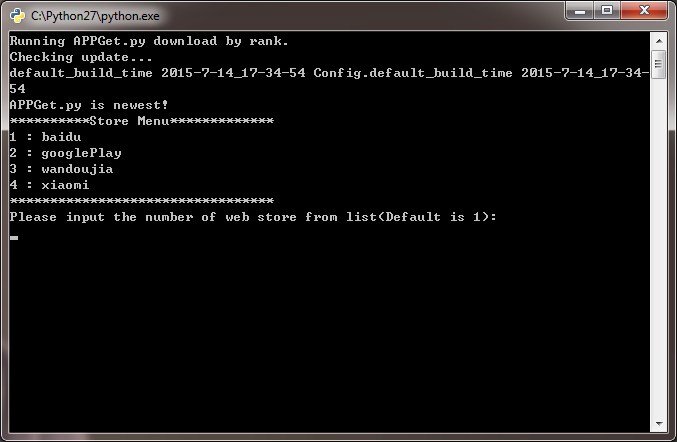
## User Guide

1. Double click or Pull into cmd.exe

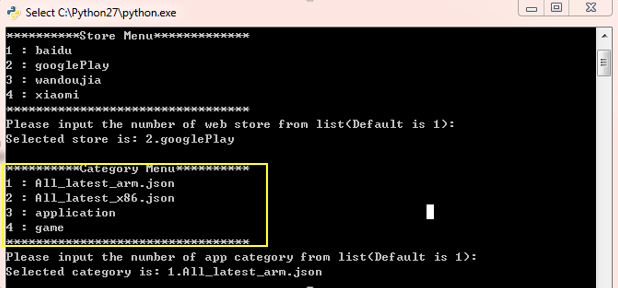
The script will be auto update by himself.



Selsect



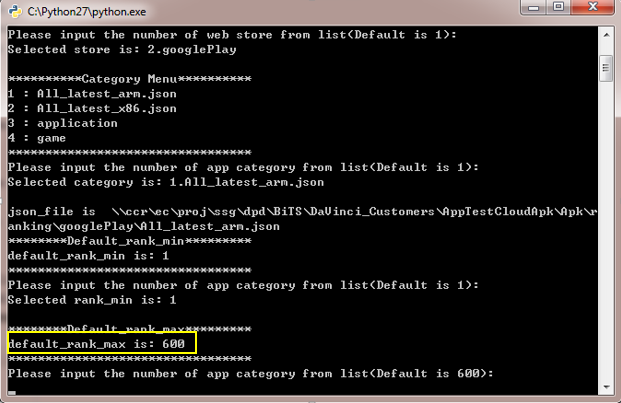
Follow guide ,firstly input the number of web store from list ,If you don’t choose the script will set default 1



Secondly, input the number of app category from list list ,If you don’t choose the script will set default 1



Whatever you choose which one such as “All\_latest\_xxxx.json” means that to downloading newest both application and game category.

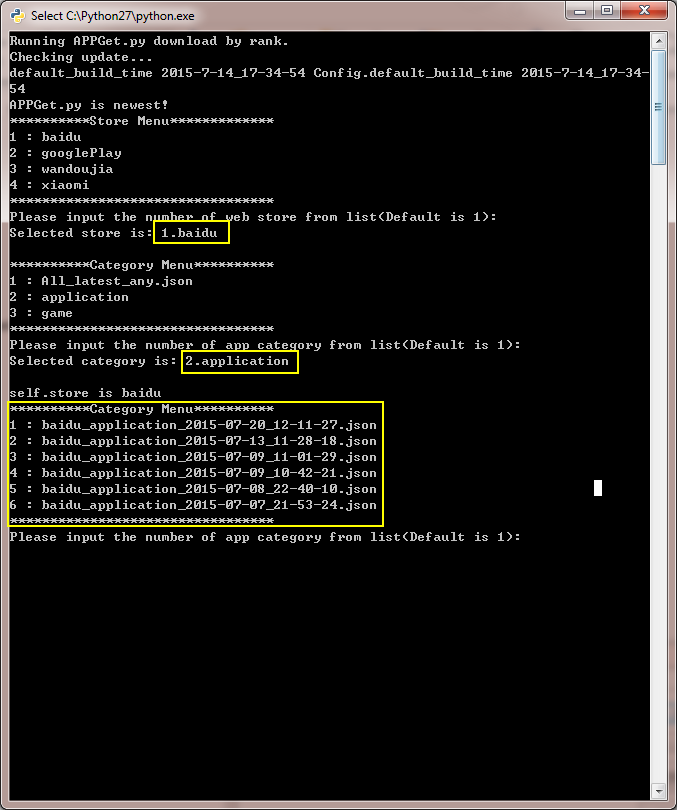


Then script will auto-load ranking form server that we provided.

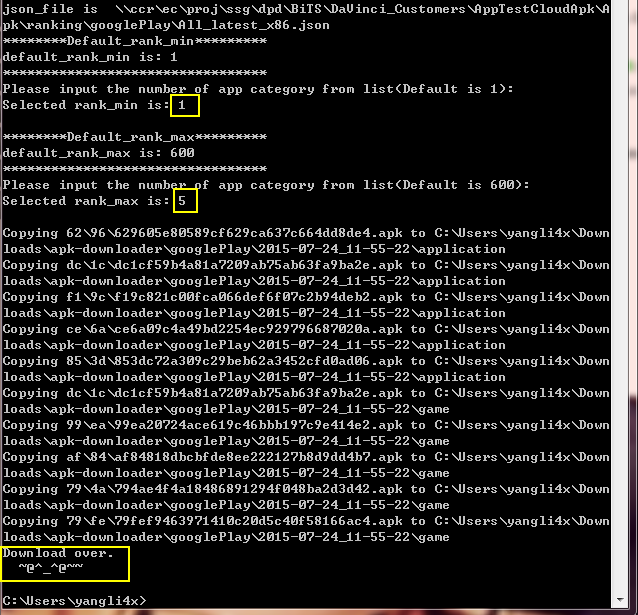
Like the picture we provided ranking max 600 from google play store each category.

1. If you want to download former version.

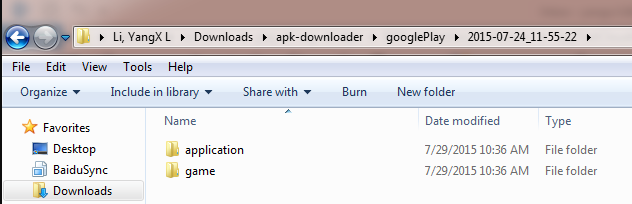
Select like this step, we will display 10 history record.

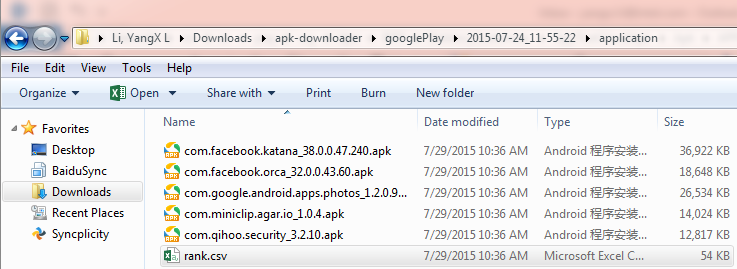


Running and end

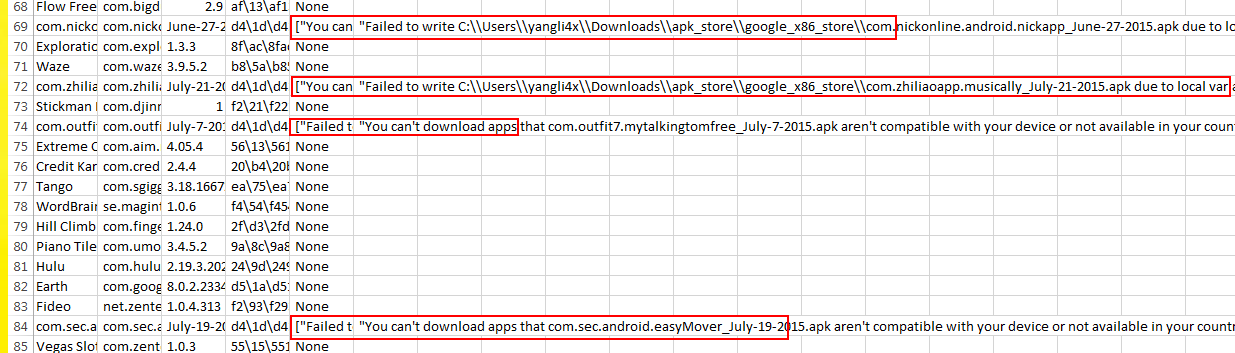


1. Result





1. When the apk download fail from web store ,we will record reason then show you on rank.csv.



# Project\_2015-8-3 Update project config parameter

## Requirements Specification

1. Let user use XML value in each PY scripts
2. Configuration.xml call such as “Config.XXXX”, GooglePlay.xml call such as “Store.XXXX”.

## Architectural Design Specification

1. Create class “Config”, This a class prepare configuration.xml variable
2. Create class “Store”, This a class prepare %STORE\_NAME%.xml variable
3. Using Setattr assignin class “Config” and “Store”.

## Procedural Design Specification

1. Class Config

class Config:

"""This a class prepare configuration.xml variable"""

1. Class Store

class Store:

"""This a class prepare %STORE\_NAME%.xml variable"""

apk\_debug\_dict = {}

1. Using “setattr”

def load(self):

*# Owner：11602272*

*# CreateTime：2015年5月12日*

*# ModifyTime：2015年5月12日*

*# 函数参数：null*

*# 函数方法：解析config.xml*

*# 函数返回值：null*

xml\_path = os.path.abspath(os.path.join(os.path.split(os.path.realpath(\_\_file\_\_))[0], os.path.pardir))

tree = ET.parse(xml\_path + os.path.sep + 'configuration.xml')

root = tree.getroot()

for child in root:

setattr(Config, child.tag, child.text)

logger.debug("%s = %s" % (child.tag, child.text), "on")

store = []

for it in root.findall("default\_store"):

store.append(it.text)

setattr(Config, "default\_store", store)

logger.debug("default\_store = %s" % store, "on")

def reload(self, store):

*# Owner：11602272*

*# CreateTime：2015年5月19日*

*# ModifyTime：*

*# 函数参数：null*

*# 函数方法：解析webStore文件夹下的web配置文件*

*# 函数返回值：null*

xml\_path = os.path.abspath(os.path.join(os.path.split(os.path.realpath(\_\_file\_\_))[0], os.path.pardir)) + os.path.sep + "webStore"

tree = ET.parse(xml\_path + os.path.sep + store + '.xml')

root = tree.getroot()

for child in root:

setattr(Store, child.tag, child.text)

logger.debug("%s = %s" % (child.tag, child.text), "on")

apk\_category = []

proxy = None

for it in root.findall("default\_apk\_category"):

apk\_category.append(it.text)

setattr(Store, "default\_apk\_category", apk\_category)

logger.debug("default\_apk\_category = %s" % apk\_category, "on")

if Store.default\_proxy\_type == "http" or Store.default\_proxy\_type == "socks":

proxy = {Store.default\_proxy\_type : Store.default\_proxy}

else:

pass

setattr(Store, "default\_proxy", proxy)

logger.debug("default\_proxy = %s" % proxy, "on")

1. In scripts

Config.default\_store

Config.default\_web\_apk\_api

Store.default\_proxy

Store.default\_download\_begin

Store.default\_apk\_save\_path

Store.default\_login\_url

## Test Plan & Report

No bug

## User Guide

None

# Project\_2015-6-15 Google Play apk downloader

## Requirements Specification

1. Download Googleplay application and game APKs
2. Follow project Project\_2015-5-08

## Architectural Design Specification

1. Base on apkCrawler.py framework.
2. Add batch login Google function

## Procedural Design Specification

1. Login and get google Token

def get\_google\_token(self):

*# Owner：11602272*

*# CreateTime：2015年6月20日*

*# ModifyTime：2015年6月25日*

*# 函数参数：登陆Google的链接，账户，密码，Proxy*

*# 函数方法：登陆Google*

*# 函数返回值：auth\_token*

account\_list = Store.default\_account.split(";")

password\_list = Store.default\_password.split(";")

for m in range(0, len(account\_list)):

params = {

"Email" : account\_list[m],

"Passwd" : password\_list[m],

"service" : "androidsecure",

"accountType" : "HOSTED\_OR\_GOOGLE"

}

headers = {"Content-type": "application/x-www-form-urlencoded"}

r = requests.post(Store.default\_login\_url, data=params, headers=headers, proxies=Store.default\_proxy)

result = r.content.split('\n')

if result[0] == 'Error=BadAuthentication':

logger.error("Login failed, please check account or password and restart the script.", "on")

auth = [i for i in result if i.find('Auth=') != -1]

if auth:

account\_token = auth[0].split('=')[1]

if account\_token != None:

self.authSubToken\_list.append(account\_token)

logger.info("Account %s authSubToken is %s" % (account\_list[m], account\_token), "on")

else:

logger.error("Get account %s token failed!" % account\_list[m], "on")

*# raise Exception("Get token failed!")*

return self.authSubToken\_list

1. Get Apk update ,assert Apk whether or not to be downloaded.

@retry(3)

def get\_update(self, package\_name):

*# Owner：11602272*

*# CreateTime：2015年7月1日*

*# ModifyTime：*

*# 函数参数：package\_name, apk\_save\_path, proxy*

*# 函数方法：取到apk最近更新时间，用来代替无法从web得到的version信息*

*# 函数返回值：update时间信息和是否download的标识*

update = time.strftime("%b-%d-%Y", time.localtime())

download\_flag = True

req = self.get\_url\_content("https://play.google.com/store/apps/details?id=%s" % package\_name)

content = req.content.decode("utf-8")

tree = lxml.html.fromstring(content)

all\_match = tree.xpath("//div[@itemprop=\"datePublished\"]")

update = str(all\_match[0].text\_content()).replace(" ", "-").replace(",", "")

expectation\_apk\_name = package\_name + '\_' + update + ".apk"

if os.path.exists(Store.default\_apk\_save\_path + os.path.sep + expectation\_apk\_name):

logger.debug("App %s already exist." % (Store.default\_apk\_save\_path + os.path.sep + expectation\_apk\_name), "on")

download\_flag = False

return update, download\_flag

1. Record Apk info

def google\_encode(self, buffer, number):

while number:

if number < 128:

mod = number

number = 0

else:

mod = number % 128

mod += 128

number = number / 128

buffer.append(mod)

def update\_data(self, buffer, data, raw=False):

if raw is False:

data\_type = type(data).\_\_name\_\_

if data\_type == "bool":

buffer.append(1 if data is True else 0)

elif data\_type == "int":

self.google\_encode(buffer, data)

elif data\_type == "str":

self.google\_encode(buffer, len(data))

for c in data:

buffer.append(ord(c))

else:

raise Exception("Unhandled data type : " + data\_type)

else:

buffer.append(data)

def generate\_request(self, para):

tmp = []

pad = [10]

result = []

header\_len = 0

url\_config = [[16], [24], [34], [42], [50], [58], [66], [74], [82], [90], [19, 82], [10], [20]]

for i in range(0, 13):

if i == 4:

self.update\_data(tmp, '%s:%d' % (para[4], para[2]))

elif i == 10:

self.update\_data(tmp, para[i])

header\_len = len(tmp) + 1

elif i == 11:

self.update\_data(tmp, len(para[i]) + 2)

else:

self.update\_data(tmp, para[i])

tmp += url\_config[i]

self.update\_data(result, header\_len)

result = pad + result + pad + tmp

stream = ""

for data in result:

stream += chr(data)

return base64.b64encode(stream, "-\_")

@retry(10)

def get\_apk\_url(self, package\_name, authSubToken\_list):

*# Owner：11602272*

*# CreateTime：2015年6月24日*

*# ModifyTime：2015年6月25日*

*# 函数参数： 需要下载的apk包名,手机的android\_id,Proxy*

*# 函数方法：null*

*# 函数返回值：req*

if authSubToken\_list:

authSubToken = authSubToken\_list[0]

authSubToken\_list.remove(authSubToken\_list[0])

else:

logger.error("authSubToken\_list is None %s download fail." % package\_name, "on")

self.package\_name\_map\_retry[package\_name] = 7

sys.exit(1)

if not self.package\_cookies.has\_key(package\_name):

update, download\_flag = self.get\_update(package\_name)

url = ""

cookies = {}

headers = {}

if download\_flag:

input\_para = [authSubToken, True, int(Store.default\_sdk\_level) , Store.default\_android\_id, "", "en", "us", "AT&T", "AT&T", "31038", "31038", package\_name, package\_name]

request = self.generate\_request(input\_para)

params = {"version" : 2, "request" : request}

headers = {"Content-type": "application/x-www-form-urlencoded",

"Accept-Language": "en\_US",

"Authorization": "GoogleLogin auth=%s" % authSubToken,

"X-DFE-Enabled-Experiments": "cl:billing.select\_add\_instrument\_by\_default",

"X-DFE-Unsupported-Experiments": "nocache:billing.use\_charging\_poller,market\_emails,buyer\_currency,prod\_baseline,checkin.set\_asset\_paid\_app\_field,shekel\_test,content\_ratings ,buyer\_currency\_in\_app,nocache:encrypted\_apk,recent\_changes",

"X-DFE-Device-Id": Store.default\_android\_id,

"X-DFE-Client-Id": "am-android-google",

"User-Agent": "Android-Finsky/3.7.13 (api=3,versionCode=8013013,sdk=16,device=crespo,hardware=herring,product=soju)",

"X-DFE-SmallestScreenWidthDp": "320",

"X-DFE-Filter-Level": "3",

"Accept-Encoding": "",

"Host": "android.clients.google.com"}

r = requests.post('https://android.clients.google.com/market/api/ApiRequest', data=params, headers=headers, verify=False, proxies=Store.default\_proxy)

if r.status\_code == 429:

logger.error("Get %s Too many request" % package\_name, "on")

elif r.status\_code == 403:

logger.error("Get %s Forbidden" % package\_name, "on")

elif r.status\_code == 401:

logger.error("Get %s Unauthorized" % package\_name, "on")

elif r.status\_code != 200:

logger.error('Unexpected status code %s' % r.status\_code, "on")

gzipped\_content = r.content

response = zlib.decompress(gzipped\_content, 16 + zlib.MAX\_WBITS)

match\_https = re.search("(https?:\/\/[^:]+)", response)

if match\_https is None:

logger.error("Get %s https failed" % package\_name, "on")

else:

url = match\_https.group(1)

match\_cookie = re.search("MarketDA.\*?(\d+)", response)

if match\_cookie is None:

logger.error("Get %s cookie failed" % package\_name, "on")

else:

cookies = {"MarketDA":match\_cookie.group(1)}

headers = { "User-Agent" : "AndroidDownloadManager/4.2.1 (Linux; U; Android 4.2.1; Galaxy Nexus Build/JRO03E)", "Accept-Encoding": "" }

else:

logger.debug("The apk %s is not update, needn't download." % package\_name, "on")

self.package\_cookies[package\_name] = [update, url, headers, cookies]

logger.info("Get %s download url, version, cookes succees!" % package\_name, "on")

else:

pass

1. GooglePlay.xml add new value

<default\_login\_url>https://android.clients.google.com/auth</default\_login\_url>

<default\_account>shwdeopenlab626@gmail.com;shwdeopenlab627@gmail.com;shwdeopenlab628@gmail.com;shwdeopenlab629@gmail.com;shwdeopenlab630@gmail.com;shwdeopenlab702@gmail.com;shwdeopenlab701@gmail.com;shwdeopenlab624@gmail.com;shwdeopenlab625@gmail.com</default\_account>

<default\_password>mys1255695864;mys1255695864;mys1255695864;mys1255695864;mys1255695864;mys1255695864;mys1255695864;mys1255695864;mys1255695864;mys1255695864</default\_password>

<default\_sdk\_level>17</default\_sdk\_level>

*<!-- <default\_android\_id>35309FB832427165</default\_android\_id> -->* *<!-- ASUS\_T00G x86\_Phone-->*

<default\_android\_id>36461642B1A70166</default\_android\_id> *<!-- Nexus 5 arm\_Phone-->*

*<!-- <default\_android\_id>3A184709549EABC6</default\_android\_id> -->* *<!-- Nexus 7 arm\_tablet-->*

*<!-- <default\_android\_id>326600F1D8118DD9</default\_android\_id> -->* *<!-- Nexus Player -->*

*<!-- <default\_arch>x86\_Phone</default\_arch> -->*

*<!-- <default\_arch>x86\_Tablet</default\_arch> -->*

<default\_arch>arm\_Phone</default\_arch>

*<!-- <default\_arch>arm\_Tablet</default\_arch> -->*

*<!-- <default\_apk\_save\_path>C:\Users\yangli4x\Downloads\apk\_store\google\_x86\_Phone\_store</default\_apk\_save\_path>*

*<default\_apk\_offline\_download\_path>C:\Users\yangli4x\Downloads\apk\_store\google\_x86\_Phone\_store</default\_apk\_offline\_download\_path> -->*

*<!-- <default\_apk\_save\_path>C:\Users\yangli4x\Downloads\apk\_store\google\_x86\_Tablet\_store</default\_apk\_save\_path>*

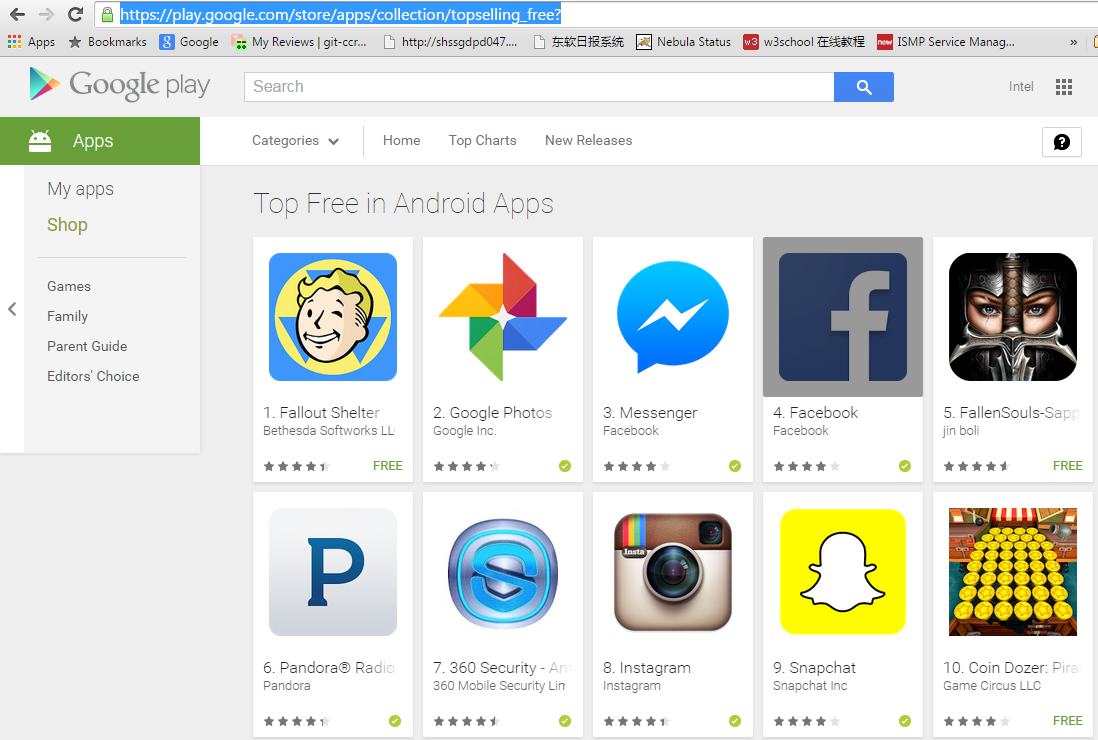
*<default\_apk\_offline\_download\_path>C:\Users\yangli4x\Downloads\apk\_store\google\_x86\_Tablet\_store</default\_apk\_offline\_download\_path> -->*

<default\_apk\_save\_path>C:\Users\yangli4x\Downloads\apk\_store\google\_arm\_Phone\_store</default\_apk\_save\_path>

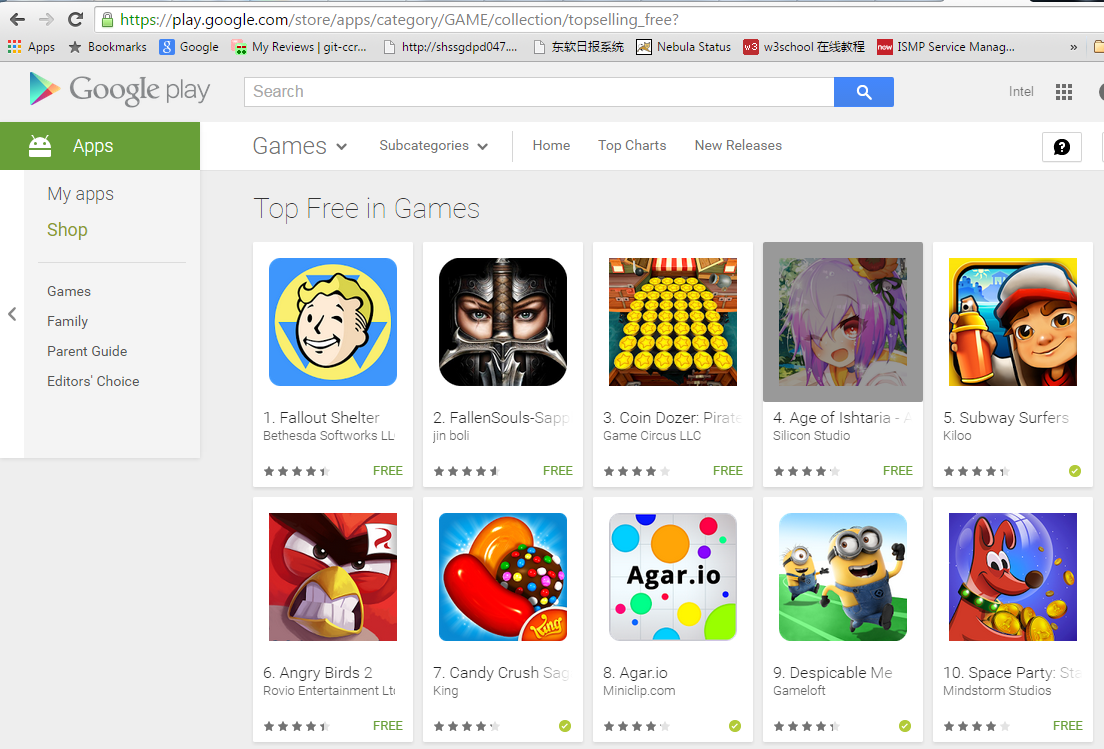
<default\_apk\_offline\_download\_path>C:\Users\yangli4x\Downloads\apk\_store\google\_arm\_Phone\_store</default\_apk\_offline\_download\_path>

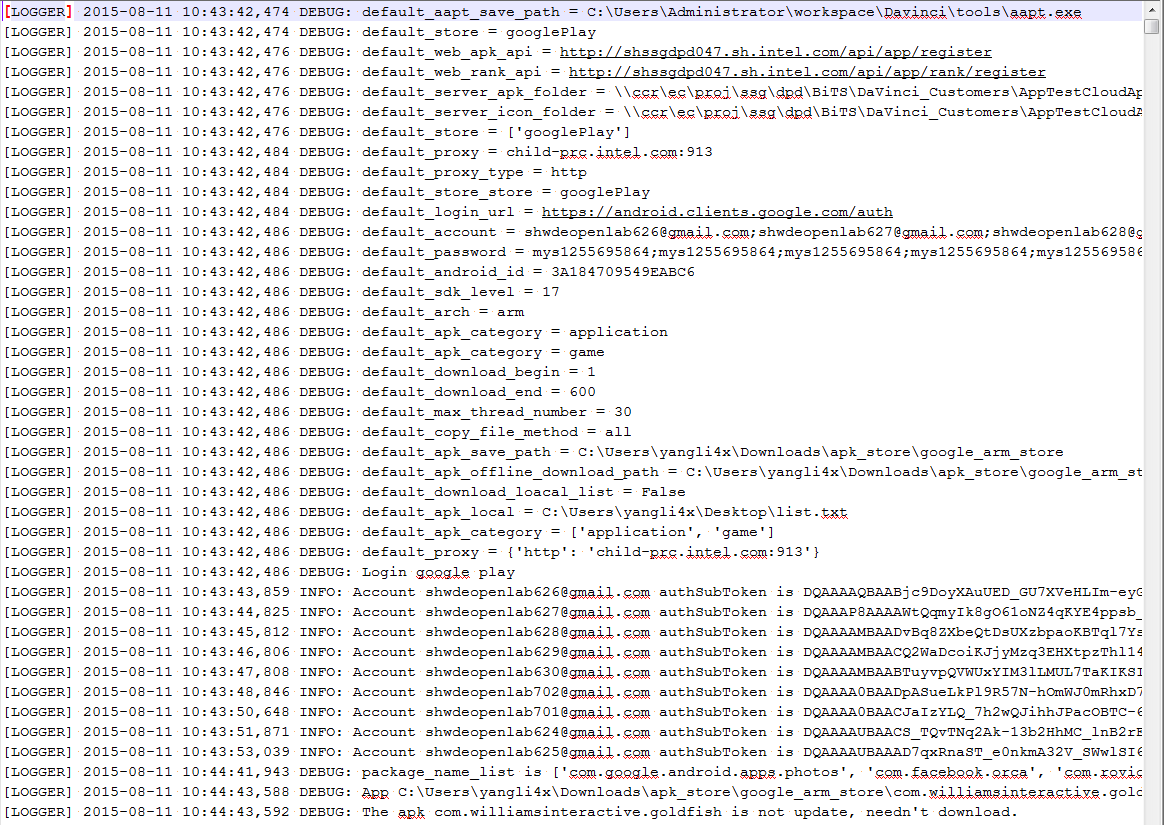
## Test Plan & Report

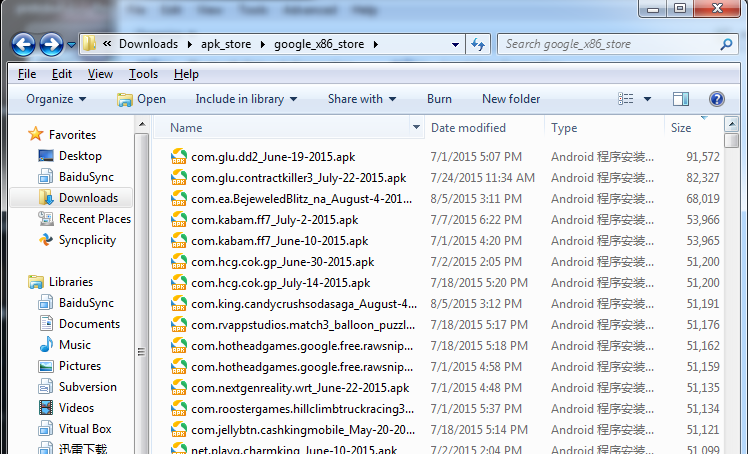
1. Google Play Application Top Free <https://play.google.com/store/apps/collection/topselling_free>?



1. Google Play Game Top Free <https://play.google.com/store/apps/category/GAME/collection/topselling_free>?



1. Running log
2. Achievement



## User Guide

Follow project Project\_2015-5-08