```
#service.txt
HD_STAR_PLUS:1:15:201
FRANCE_24:1:15:202
BINDASS:1:15:203
HUNGAMA:1:15:204
24_GHANTE:1:15:205
DISNEY_JUNIOR:1:15:206
HD_ZEE_TV:1:15:207
KOLKATA_TV:1:15:208


#! /bin/bash
#startEPG.sh
#for updating the time
runningcount=`pgrep -c 'startEPG.sh'`
if [ "${runningcount}" -eq "2" ];then
    echo -e "\n\e[1;31mEPG sending already running\e[0m\n"
    exit 0
fi
myexe=${0}
mypwd=`dirname ${myexe}`
cd ${mypwd}
version_number=-1
debug="1"
while [ 1 ]
do
    let version_number=$version_number+1
    while IFS=: read serviceName transportId eventsPerSection newServiceId
    do

        if [ $version_number -gt 30 ];then
            version_number=0
        fi
        printf "version_number:"$version_number"\n"
        echo "serviceName: $serviceName :transportId : $transportId eventsPerSection :
$eventsPerSection newServiceId : $newServiceId"
    python ${mypwd}/src/UpdateDBTime.py  ${serviceName}
    python ${mypwd}/src/EpgSectionGenerator.py $serviceName $transportId $eventsPerSection
$newServiceId $version_number
    python ${mypwd}/src/EpgTDT.py $version_number
        sec2ts 18 < ${mypwd}/$serviceName'.sec' > ${mypwd}/$serviceName'.ts'
        sec2ts 20 < ${mypwd}/tdt.sec > ${mypwd}/tdt.ts
        dvbsnoop -if ${mypwd}/$serviceName'.sec' -ph 0 -pd 9 -s sec -tssubdecode | grep event_name: |
wc -l
        dvbsnoop -if ${mypwd}/$serviceName'.ts' -ph 0 -pd 9 -s ts -tssubdecode | grep  event_name: |
wc -l
    if [ "${debug}" -eq "1" ];then
        tsudpsend ${mypwd}/$serviceName'.ts' 224.0.1.99 1234 40000
        tsudpsend ${mypwd}/tdt.ts 224.0.1.100 1234 20000
    fi
    if [ "${debug}" -eq "1" ];then
        rm ${mypwd}/tdt.sec ${mypwd}/tdt.ts
        rm ${mypwd}/$serviceName'.sec'
        rm ${mypwd}/$serviceName'.ts'
    fi
    done < ${mypwd}/service.txt
done


#reference from internet
#while IFS=: read user pass uid gid full home shell
#do
#echo -e "$full :\n
# Pseudo : $user\n
# UID :\t $uid\n
# GID :\t $gid\n
# Home :\t $home\n
# Shell :\t $shell\n\n"
#done < /etc/passwd
```

```
#!/usr/bin/python
#EpgSectionGenerator.py

# <author>Pieter Muller</author>
# <date>2013-03-05</date>
# <note>Targets Python 2.7</note>

import time
from time import gmtime
import sqlite3 as sqlite
from xml.dom import minidom
import sys
from EPGSection import *

def getServiceId(cursor):
    for serviceId in cursor.execute('SELECT DISTINCT serviceId FROM present_services;'):
        return serviceId

def getRowCount(cursor):
    for rowCount in cursor.execute('SELECT rowCount FROM epgEventRows;'):
        return rowCount

def populateDB(scriptfilename,dbfilename,servicename):
    try:
        #print "\nOpening DB"
        connection = sqlite.connect(dbfilename)
        cursor = connection.cursor()
        #print "Reading Script..."
        scriptFile = open(scriptfilename, 'r')
        script = scriptFile.read()
        scriptFile.close()
        #print "Running Script..."
        fromDateTime=int(round(time.time()))
        script=script.replace("SERVICE_NAME", servicename)
        script=script.replace("FROM_DATETIME", str(fromDateTime))
        cursor.executescript(script)
        connection.commit()
        #print "Changes successfully committed\n"

        serviceId=getServiceId(cursor)
        rowCount=getRowCount(cursor)
    except Exception, e:
        print "Something went wrong:"
        print e
    finally:
        #print "\nClosing DB"
        connection.close()
        #print "\nDB Closed"
        return serviceId,rowCount

if __name__ == "__main__":
    if (len(sys.argv) < 5):
        print "\n\tRequires at least four arguments:"
        print "\n\t\EpgSectionGenerator.py {serviceName}{transportStreamId}{eventsPerSection}
{newServiceId}{version_number}\n\n"
        sys.exit()

    serviceName=sys.argv[1]
    serviceName=serviceName.replace("_", " ")
    transportStreamId=int(sys.argv[2])
    eventsPerSection=int(sys.argv[3])
    newServiceId=int(sys.argv[4])
    versionNumber=int(sys.argv[5])
    originalNetworkId=0

    #populate database for current service
    myexe=sys.argv[0]
    mypwd=str(os.path.dirname(myexe))+'/'
    scriptFileName=mypwd + '../db/present_services.sql'
    aeondbfile=mypwd + '../db/aeon_epg.db'
    serviceId,rowCount=populateDB(scriptFileName, aeondbfile,serviceName)
    print "\t\t", serviceName, serviceId[0], rowCount[0]
```

```
    #create sections
    epgSections=EPGSection(int(serviceId[0]), int(transportStreamId), originalNetworkId, int(rowCount
[0]),  serviceName, eventsPerSection, newServiceId,versionNumber)
    epgSections.createSchedSections()

    #------------------------------- Create Section Files Using XML
-----------------------------------
    #eventsPerSection=int(sys.argv[2])
    #xmldoc = minidom.parse(servicetsidfilename)
    #itemlist = xmldoc.getElementsByTagName('service')

    #for element in itemlist :
    #    print "Service : ", element.attributes['name'].value, "TsId : ", element.attributes
['tsid'].value
    #    serviceName=element.attributes['name'].value
    #    transportStreamId=element.attributes['tsid'].value
    #    originalNetworkId=0
    #    serviceId,rowCount=populateDB('present_services.sql', 'aeon_epg.db', serviceName)
    #    epgSections=EPGSection(int(serviceId[0]),int(transportStreamId),originalNetworkId, int
(rowCount[0]), serviceName,eventsPerSection)
    #    epgSections.createSections()

#!/usr/bin/python
#src/EPGSection.py
import os
import sys
#sys.path.append(os.environ["OPENCASTER_LIB"])
import sqlite3
import string
from dvbobjects.PSI.PAT import *
from dvbobjects.PSI.NIT import *
from dvbobjects.PSI.SDT import *
from dvbobjects.PSI.TDT import *
from dvbobjects.PSI.EIT import *
from dvbobjects.PSI.PMT import *
from dvbobjects.DVB.Descriptors import *
from dvbobjects.MPEG.Descriptors import *
from EventInfo import *
myexe=sys.argv[0]
mypwd=str(os.path.dirname(myexe)) + '/'
class EPGSection(object):
    serviceName=""
    serviceId=0
    transportStreamId=0
    originalNetworkId=0
    rowCount=0
    descriptorLoop=[]
    eventItem=()
    eventLoopItems=[]
    sectionFd=0
    eventsPerSection=1
    newServiceId=0
    versionNumberPF=0
    versionNumberSCHED=0
    def __init__(self,serviceId,transportStreamId,originalNetworkId, rowCount,
serviceName,eventsPerSection,newServiceId,versionNumber):
        self.serviceName=serviceName
        self.serviceId=serviceId
        self.transportStreamId=transportStreamId
        self.originalNetworkId=originalNetworkId
        self.rowCount=rowCount
        self.descriptorLoop=[]
        self.eventItem=()
        self.eventLoopItems=[]
        self.sectionFd=0
        self.eventsPerSection=eventsPerSection
        self.newServiceId=newServiceId
        self.versionNumberPF=versionNumber
        self.versionNumberSCHED=versionNumber
    def getEventLoopItem(self,event):
        self.descriptorLoop.append(short_event_descriptor (
                ISO639_language_code = event.ISO639_language_code,
```

```
                event_name = str(event.event_name),
                text = str(event.shortDescText),
                )
        )
        self.descriptorLoop.append(content_descriptor(
                content_user_loop=[
                    content_user_loop_item(
                    content_nibble_level_1=int(event.metaInfo[21]),
                    content_nibble_level_2=int(event.metaInfo[43]),
                    user_nibble_level_1=0x00,
                    user_nibble_level_2=0x00,
                    ),
                ]
            )
        )

        bytesPerDescriptor=250
        mText=str(filter(lambda x: x in string.printable, event.longDescText))
        mlast_descriptor_number = len(mText)/bytesPerDescriptor
        for i in range(mlast_descriptor_number + 1):
            self.descriptorLoop.append(extended_event_descriptor(
                descriptor_number=i,
                last_descriptor_number=mlast_descriptor_number,
                ISO639_language_code=event.ISO639_language_code,
                extended_event_loop=[],
                text=mText[i * bytesPerDescriptor: (i+1) * bytesPerDescriptor - 1]
                )
            )

        self.eventItem=event_loop_item(
            event_id = event.event_id,
            start_year = event.start_year, # since 1900
            start_month = event.start_month,
            start_day = event.start_day,
            start_hours = event.start_hours,
            start_minutes = event.start_minutes,
            start_seconds = event.start_seconds,
            duration_hours = event.duration_hours,
            duration_minutes = event.duration_minutes,
            duration_seconds = event.duration_seconds,
            running_status = 4, # 4 service is running, 1 not running, 2 starts in a few seconds, 3
pausing
            free_CA_mode = 0, # 0 means service is not scrambled, 1 means at least a stream is
scrambled
            event_descriptor_loop = self.descriptorLoop,
        )
        self.descriptorLoop=[]

    def write(self, currentSectionNumber, lastSectionNumber, tableId,versionNumber=1001):
        eit = event_information_section(
                table_id = tableId,
                last_table_id = EIT_ACTUAL_TS_SCHEDULE58,
                service_id = self.newServiceId,
                transport_stream_id = self.transportStreamId,
                original_network_id = self.originalNetworkId,
                event_loop = self.eventLoopItems,
                segment_last_section_number = currentSectionNumber,
                version_number =  versionNumber,
                section_number = currentSectionNumber,
                last_section_number = lastSectionNumber,   # pay attention here, we have another
section after this!
                )
        self.outFd.write(eit.pack())
        self.eventLoopItems=[]

    def createSchedSections(self):
        sectionfile=mypwd + '../' + self.serviceName.replace(" ", "_")+'.sec'
        self.outFd = open(sectionfile, "wb")

        currentRow=0
        currentSectionNumber=0
        lastSectionNumber=self.rowCount/self.eventsPerSection
```

```
        if (self.rowCount % self.eventsPerSection) == 0:
            lastSectionNumber=lastSectionNumber-1

        lastSectionNumber=lastSectionNumber+1           #we are going to add section for
present_following

        eitDBConnection = sqlite3.connect(mypwd + '../db/aeon_epg.db')
        eitDBCursor = eitDBConnection.cursor()
        for eventInfo in eitDBCursor.execute("select * from present_services order by starttime;"):
            event=EventInfo(eventInfo);
            self.getEventLoopItem(event)
            self.eventLoopItems.append(self.eventItem)
            currentRow=currentRow+1
            if (currentRow % self.eventsPerSection) == 0:
                self.write(currentSectionNumber, lastSectionNumber,
EIT_ACTUAL_TS_SCHEDULE58,self.versionNumberSCHED)
                currentSectionNumber=currentSectionNumber+1
        eitDBCursor.close()
        if (currentRow % self.eventsPerSection) > 0:
            self.write(currentSectionNumber, lastSectionNumber,
EIT_ACTUAL_TS_SCHEDULE58,self.versionNumberSCHED)
        self.createPFSections(lastSectionNumber)

    def createPFSections(self, lastSectionNumber):
        currentSectionNumber=lastSectionNumber
        eitDBConnection = sqlite3.connect( mypwd + '../db/aeon_epg.db')
        eitDBCursor = eitDBConnection.cursor()
        for eventInfo in eitDBCursor.execute("select * from present_services_pf order by starttime;"):
            event=EventInfo(eventInfo);
            self.getEventLoopItem(event)
            self.eventLoopItems.append(self.eventItem)
            print "\t-", event.event_name
        self.write(currentSectionNumber, lastSectionNumber,
EIT_ACTUAL_TS_PRESENT_FOLLOWING,self.versionNumberPF)

        eitDBCursor.close()
        self.outFd.close

#!/usr/bin/python
#src/EpgTDT.py
import os
import sys
#sys.path.append(os.environ["OPENCASTER_LIB"])
import time
from time import gmtime
from dvbobjects.PSI.PAT import *
from dvbobjects.PSI.NIT import *
from dvbobjects.PSI.SDT import *
from dvbobjects.PSI.TDT import *
from dvbobjects.PSI.EIT import *
from dvbobjects.PSI.PMT import *
from dvbobjects.DVB.Descriptors import *
from dvbobjects.MPEG.Descriptors import *
#
# Time Description Table (ETSI EN 300 468 5.2.5)
# it should be replaced at run time with tstdt
#
myexe=sys.argv[0]
mypwd=str(os.path.dirname(myexe))+'/'

class TimeDate(object):
    year = 0 # since 1900
    month = 0
    day = 0
    hour = 0# use hex like decimals
    minute = 0
    second = 0
    versionNumber=0
    def __init__(self,versionNumber):
        self.year = 0 # since 1900
        self.month = 0
        self.day = 0
```

```
        self.hour = 0 # use hex like decimals
        self.minute = 0
        self.second = 0
        self.versionNumber=versionNumber
    def getTime(self,var):
        return int(time.strftime(var,gmtime()))

    def getBCD(self, x):
        return (x/10 * 16 + x % 10)

    def setInfo(self):
        self.year = self.getTime("%Y")-1900 # since 1900
        self.month = self.getTime("%m")
        self.day = self.getTime("%d")
        self.hour = self.getBCD(self.getTime("%H"))  # use hex like decimals
        self.minute = self.getBCD(self.getTime("%M")) # use hex like decimals
        self.second = self.getBCD(self.getTime("%S")) # use hex like decimals

    def create_time_date_section(self):
        tdt = time_date_section(
            year = self.year, # since 1900
            month = self.month,
            day = self.day,
            hour = self.hour , # use hex like decimals
            minute = self.minute ,
            second = self.second,
            version_number = 1234,#self.versionNumber
            section_number = 0,
            last_section_number = 0,
            )
        secfile=mypwd + '../tdt.sec'
        out = open(secfile, "wb")
        out.write(tdt.pack())
        out.close
        out = open(secfile, "wb") # python   flush bug
        out.close
        #os.system('/usr/local/bin/sec2ts 20 < ./tdt.sec > ./tdt.ts')

versionNumber=sys.argv[1]
time_date=TimeDate(versionNumber)
time_date.setInfo()
time_date.create_time_date_section()

#!/usr/bin/python
#src/EPGSection.py
import time
class EventInfo(object):
    service_id=0
    transport_stream_id=0
    event_id=0
    start_year=0
    start_month=0
    start_day=0
    start_hours=0
    start_minutes=0
    start_seconds=0
    duration_hours=0x23
    duration_minutes=0x00
    duration_seconds=0x00
    running_status=4
    free_CA_mode=0
    ISO639_language_code=" "
    event_name=" "
    shortDescText=" "
    longDescText=" "
    version=0
    eventInfo=" "
    metaInfo=" "

    def getTime(self):
        return time.gmtime(self.eventInfo[0][7])
```

```
    def getBCD(self, x):
        return (x/10 * 16 + x % 10)

    def getLanguage(self):
        iso639code=self.eventInfo[0][12]
        c1=iso639code & 0x0000FF
        c2=iso639code & 0x00FF00
        c2>>=8
        c3=iso639code & 0xFF0000
        c3>>=16
        return (chr(c1) + chr(c2) + chr(c3))

    def __init__(self,eventInfo):
        self.eventInfo=[eventInfo];
        self.service_id=self.eventInfo[0][4]
        self.transport_stream_id=self.eventInfo[0][3]
        self.event_id=self.eventInfo[0][5]
        mLocalTime=self.getTime()

        self.start_year=(mLocalTime.tm_year-1900)
        self.start_month=mLocalTime.tm_mon
        self.start_day=mLocalTime.tm_mday
        self.start_hours=self.getBCD(mLocalTime.tm_hour)
        self.start_minutes=self.getBCD(mLocalTime.tm_min)
        self.start_seconds=self.getBCD(mLocalTime.tm_sec)

        self.metaInfo=self.eventInfo[0][15]

        duration=self.eventInfo[0][8];
        self.duration_hours=int(duration/(3600))
        duration=duration-self.duration_hours*3600
        self.duration_minutes=int(duration/60)
        self.duration_seconds=int(duration%(60))

        self.duration_hours = self.getBCD(self.duration_hours)
        self.duration_minutes = self.getBCD(self.duration_minutes)
        self.duration_seconds = self.getBCD(self.duration_seconds)

        self.running_status=4
        self.free_CA_mode=0
        self.ISO639_language_code=self.getLanguage()
        self.event_name=self.eventInfo[0][16]

        tempText=self.eventInfo[0][17]
        self.shortDescText=tempText
        tempText=self.eventInfo[0][18]
        self.longDescText=tempText

        self.version=self.eventInfo[0][20]

        #print "Event :",self.event_name, "Start Time: ", self.start_hours, ":", self.start_minutes
        #print "Event :",self.event_name, "Duration  : ", self.duration_hours, ":",
self.duration_minutes

#!/usr/bin/python
#src/UpdateDBTime.py
import sys
import os
import time
import sqlite3 as sqlite
def CheckValidDBTime():
    try:
        myexe=sys.argv[0]
        mypwd=str(os.path.dirname(myexe)) + '/'
        dbfile= mypwd + "../db/epg_sched.db"
        connection = sqlite.connect(dbfile)
        cursor = connection.cursor()
        nowTime=int(time.time())
        servicename=sys.argv[1]
        servicename.replace("_", " ")
        settings="attach " + "'" + mypwd  + "../db/settings.db"+ "'" + " as settings;"
        epg="attach " + "'" + mypwd  + "../db/epg_sched.db"+ "'" + " as epg;"
```

```
        cursor.execute(settings);
        cursor.execute(epg);
        query=("select distinct(service.serviceId) from settings.service, epg.epg where service.name
like %r and epg.serviceId=service.serviceId;")%(servicename)
        cursor.execute(query);
        serviceId=cursor.fetchone()
        serviceId=int(serviceId[0])
        query=("select count(epg.startTime) from settings.service, epg.epg where service.name like %r
and epg.serviceId=service.serviceId and epg.startTime > %r")%(servicename , nowTime)
        cursor.execute(query);
        result=cursor.fetchone()
        result=int(result[0])
    except Exception, e:
        print "Something went wrong in UpdateDBTime.py (CheckValidDBTime)"
        return
    finally:
        connection.close()

    #print serviceId ,result
    if result == 0 :
        UpdateDB(dbfile, serviceId)
    else:
        return

def UpdateDB(dbfile, serviceId):
    try:
        print "Time updating...Plase wait. \n", serviceId
        connection = sqlite.connect(dbfile)
        cursor = connection.cursor()
        query=("select min(startTime) from epg where serviceId like %r;")%(serviceId)
        cursor.execute(query)
        minStartTime=cursor.fetchone()
        minStartTime=int(minStartTime[0])
        timeDiff=round(time.time()) - minStartTime
        timeDiff=(timeDiff+19800)-(86400+1800)
        query=("update epg SET startTime=startTime + %r,endTime=endTime + %r where serviceId like %r;")
%(timeDiff,timeDiff, serviceId)
        cursor.execute(query);
        connection.commit()
    except Exception, e:
        print "Something went wrong in UpdateDBTime.py (UpdateDB)"
        print e
    finally:
        print "Time updated successfully ...\n"
        connection.close()
CheckValidDBTime();

#ref
    #select count(startTime) from epg where startTime > 1438253475 and serviceId=518;
    #select count(epg.startTime) from settings.service, epg.epg where service.name like 'Zee Cinema'
and epg.serviceId=service.serviceId and epg.startTime > 1438254638;
```