CDR -

CDR (Python), , TCP- Oracle. CDR Asterisk PBX. .

, , - . CDR .

cdrd.py ()

```
#!/usr/bin/env python

import daemon
import socket
import re
import os
import sys
import select
import signal
import threading
import cx_Oracle

from time import strftime, sleep
from datetime import datetime

os.environ["ORACLE_HOME"]="/usr/lib64/oracle/10.2.0.3/client"
os.environ["TNS_ADMIN"]="/etc/oracle"
os.environ["NLS_LANG"]="RUSSIAN_CIS.AL32UTF8"

bind_ip="127.0.0.1"
bind_port=50000

db_login="AIS_TEL"
db_password="my_secret_password"
db_inst="sl"

pid_file="/tmp/cdrd.pid"
log_file="/tmp/cdrd.log"

class Server:
        def __init__(self):
                self.host = bind_ip
                self.port = bind_port
                self.backlog = 1024
                self.size = 1024
                self.server = None
                self.threads = []
                self.db_login = db_login
                self.db_password = db_password
                self.db_inst = db_inst
                self.log = sys.stdout
#                self.log = open(log_file,'a+')
                self.pid_file = pid_file
                self.connection = None
                self.select_timeout = 180
                self.input = [];

                signal.signal(signal.SIGTERM, self.sig_handler)
                signal.signal(signal.SIGINT, self.sig_handler)
                signal.signal(signal.SIGHUP, self.sig_handler)

        def sig_handler(self,signum, frame):
                self.logger("Got signal "+str(signum))
                if signum == signal.SIGTERM or signum == signal.SIGINT:
                        self.exit(0)
                if signum == signal.SIGHUP:
                        self.reload(0)

        def reload(self,code):
                if self.connection:
```

```
                        self.logger("Closing database connection...")
                        self.connection.close()
                        self.connection = None

                self.open_oracle()

        def exit(self,code):
                self.logger("Closing database connection...")
                if self.connection: self.connection.close()
                self.logger("Closing socket...")
                if self.server: self.server.close()
                self.logger("Waiting for threads...")
                for c in self.threads:
                        c.join()
                self.logger("Exiting...")
                sys.exit(code)

        def logger(self, message):
                print >>self.log, strftime("%Y-%m-%d %H:%M:%S")+" hcdrd:", message
                self.log.flush()

        def open_socket(self):
                while not self.server:
                        try:
                                self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                                self.server.bind((self.host,self.port))
                                self.server.listen(self.backlog)
                        except socket.error, (value,message):
                                if self.server:
                                        self.server.close()
                                self.logger("Could not open socket: "+message)
                                self.logger("Retrying in 10 sec")
                                self.server = None
                                sleep(10)

                self.logger("Listening on "+str(self.host)+" port "+str(self.port))
                self.input.append(self.server)

        def open_oracle(self):
                self.logger("Trying to connect to database... ")
                try:
                        self.connection=cx_Oracle.connect( user = self.db_login,\
                                                           password = self.db_password,\
                                                           dsn = self.db_inst,\
                                                           threaded = True)
                except cx_Oracle.DatabaseError,info:
                        self.logger("Logon Error on "+str(self.db_inst)+": "+str(info).rstrip())
                        self.connection = None
                        return False

                self.logger("Successfull login to database "+self.db_inst)


        def accept(self):
                try:
                        inputready,outputready,exceptready = select.select(self.input,[],[],self.
select_timeout)
                except select.error,msg:
                        if msg[0] == 4:
                                return True
                        else:
                                self.logger("Select error: "+str(msg[1]))
                                return False

                for s in inputready:
                        if s == self.server:
                                if self.connection:
                                        c = PostCDR(self.server.accept(),self.connection.cursor(),self.logger)
                                else:
                                        c = PostCDR(self.server.accept(),None,self.logger)
                                c.start()
```

```
                                self.threads.append(c)

        def run(self):
#                daemon.daemonize(self.pid_file)

                self.logger("||| Server started |||")

                self.open_socket()

                running = 1

                while running:
                        if self.connection:
                                try:
                                        self.connection.ping()
                                except cx_Oracle.Error,info:
                                        self.logger("Connection to database is dead: "+str(info).rstrip())
                                        self.connection = None
                                        self.open_oracle()
                        else:
                                self.open_oracle()

                        self.accept()

class PostCDR(threading.Thread):
        def __init__(self,(client,address),cursor,logger):
                threading.Thread.__init__(self)
                self.client = client
                self.address = address
                self.size = 1024
                self.cursor = cursor
                self.cdr = {}
                self.logger = logger

        def postcdr(self):
                if self.cursor:
                        try:
                                self.cursor.callproc("""AIS_NET.EX_AAA_PKG.CDR_PUT""",\
                                        ("Asterisk", \
                                        "CDR_Status_Finished", \
                                        str(self.cdr['sid']), \
                                        None, \
                                        1094, \
                                        None, \
                                        None, \
                                        None, \
                                        None, \
                                        str(self.cdr['calling_id']), \
                                        str(self.cdr['called_id']), \
                                        str(self.cdr['route_a']), \
                                        str(self.cdr['route_b']), \
                                        datetime.strptime(str(self.cdr['start']), "%Y-%m-%d %H:%M:%S"), \
                                        datetime.strptime(str(self.cdr['end']), "%Y-%m-%d %H:%M:%S"), \
                                        int(self.cdr['duration']), \
                                        None, \
                                        None, \
                                        None, \
                                        str(self.cdr['cause']), \
                                        1, \
                                        "Uploaded via cdrd.py"))

                        except cx_Oracle.Error,info:
                                self.logger("Database error: "+str(info).rstrip())
                                return False
                        except ValueError,msg:
                                self.logger("Value error: "+str(msg))
                                return False
                        except Exception,msg:
                                self.logger("Unknown error: "+str(msg))
                                return False
```

```
                        return True
                else:
                        return

        def readline(self):
                buffer = self.client.recv(self.size)
                done = False

                while not done:
                        if "\n" in buffer:
                                (line, buffer) = buffer.split("\n", 1)
                                yield line.rstrip()
                        else:
                                more = self.client.recv(self.size)
                                if not more:
                                        done = True
                                        self.client.close()
                                else:
                                        buffer = buffer+more

                if buffer:
                        yield buffer.rstrip()

        def run(self):
                running = 1
                strings = iter(self.readline())

                for key in ("sid", "start", "end", "calling_id", "called_id", "route_a", "route_b", "duration",
"cause"):
                        try:
                                data = strings.next()
                        except:
                                self.logger("Connection closed unexpectedly with "+str(self.address[0])+":"+str
(self.address[1]))
                                return 1

                        self.cdr[key] = data

                result = self.postcdr()

                if result:
                        self.client.send("200 OK\n")
                else:
                        self.client.send("700 ERROR\n")

                self.client.close()


if __name__ == "__main__":
        print "Pidfile is",pid_file
        print "Logging to",log_file
        print "Starting..."

        s = Server()
        s.run()
```