

ООО „КРЮГЕР ХАУС“

Документация по блоку обмена между Artix и УНФ в части кассовых смен  
*Конфигурация УНФ 3.0*



Новосибирск, 2023 г.



# Оглавление

	Стр
1 <b>Установка и подготовка к использованию</b> . . . . .	1
1.1    Начальные (текущие) системные требования . . . . .	1
1.2    Установленные пакеты . . . . .	1
1.3    Установка . . . . .	1
1.4    Настройка запуска по расписанию . . . . .	2
2 <b>Алгоритм</b> . . . . .	5
3 <b>Установка ПО</b> . . . . .	9



## 1 Установка и подготовка к использованию

### 1.1 Начальные (текущие) системные требования

- \* Процессор - Intel(R) Atom(TM) CPU D2500 @ 1.86GHz, 2 ядра
- \* ОЗУ - 2 Гб
- \* SSD - 120 Гб
- \* Операционная система (не ниже) - Ubuntu 22.04 jammy

### 1.2 Установленные пакеты

- \* Python версии  $\geq 3.11$
- \* Webmin (не обязательно) - панель для администрирования сервера
- \* OpenSSH (установка 3.0.1.)
- \* редактор Nano (установка 3.0.2. )

### 1.3 Установка

- Копируем каталог «Workshift\_load» с программой на рабочий сервер
- Заходим в каталог «src» и выполняем команду

```
chmod +x wsh_load.py
```

Для того, что бы сделать файл скрипта исполняемым, в противном случае он не будет запускаться.

- Переходим в корневой каталог

```
cd ..
```

- В корневом каталоге выполняем команду для создания виртуального окружения

```
python3.11 -m venv .venv
```

- Выполняем команду для установки нужных пакетов

```
pip install -r requirements.txt
```

## 1.4 Настройка запуска по расписанию

- Выполняем команду для запуска планировщика

```
crontab -e
```

- В открывшемся редакторе в конец файла добавить строку

```
* * * * * /home/administrator/Workshift_load/src/wsh_load.py $HOME/command.log 2>&1
```

Здесь мы указываем периодичность запуска, полный путь к исполняемому скрипту, путь до файла в который будут выводиться сообщения планировщика. Сохранить файл. Теперь при таких настройках скрипт будет запускаться каждую минуту. Для работы нужно выставить нужный интервал запуска. Он выставляется в первой секции строки:

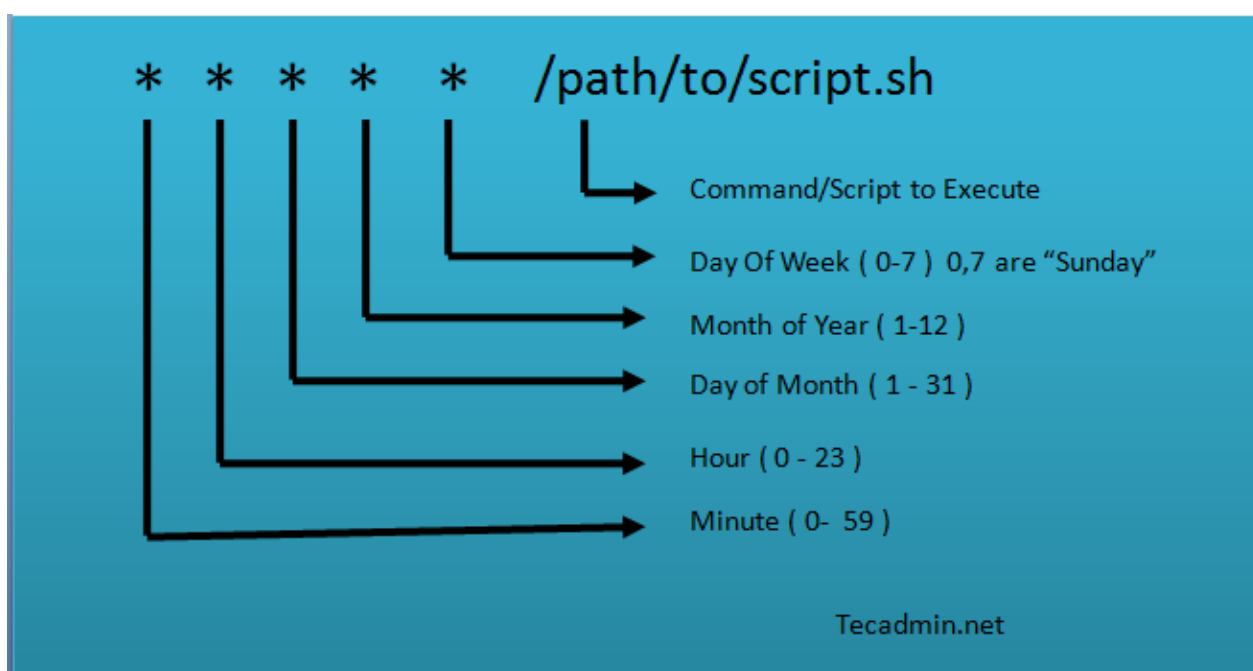


Рисунок 1.1 – «Формат Linux Crontab».

Синтаксис: `*(Minute)*(Hour)*(Day of the Month)*(Month of the Year) *(Day of the Week) username <path to command/script to execute>`

Минуты	Это значение может быть в пределах 0 — 59
Часы	Это значение может быть в пределах 0 — 23
День месяца	Это значение может быть в пределах 1 — 31
Месяц в году	Это значение поля находится в диапазоне от 1 до 12. Так же можно использовать три первые буквы названия месяца, например: jan, feb, mar
День недели	Это значение поля находится в диапазоне от 0 до 7. Где 0 и 7-воскресенье. 1-понедельник, 2-вторник и так далее

### Пример:

Следующее выражение для выполнения задачи каждые 5 минут.

```
*/5 * * * * /home/administrator/Workshift_load/src/wsh_load.py
```

## Структура проекта





## 2 Алгоритм

- При запуске программы производится проверка на наличие в каталоге «src» файла «first.dat». Если он не обнаружен это считается первым запуском программы и вызывается процедура инициализации, которая создает два текстовых файла с датами равными началу года. Таким образом при первом запуске по запросу с этими заведомо ранними датами, в результат запроса попадут все кассовые смены имеющиеся в Artix, а даты в файлах будут установлены на максимальное значение открытия и закрытия смен.

```
def init_pr():

    filename = '/home/administrator/Workshift_load/src/last_date.txt'
    if not os.path.exists(filename):
        with open(filename, 'w', encoding='utf-8') as outfile:
            outfile.write('2023-01-01 00:00:00')

    filename = '/home/administrator/Workshift_load/src/last_date_open.txt'
    if not os.path.exists(filename):
        with open(filename, 'w', encoding='utf-8') as outfile:
            outfile.write('2023-01-01 00:00:00')
```

Листинг 1 – Python example

- При запуске программы происходит чтение настроек. Если чтение успешно, запускается функция «main()», в противном случае программа завершает свою работу.

```
if __name__ == "__main__":

    # Чтение настроек
    m_conf = m_config.m_Config()
    rc = m_conf.loadConfig()
    if not rc == None:
        main()
    else:
        logger.info(u'Программа завершила работу')
```

Листинг 2 – Python example

Настройки хранятся в файле /config/config.ini . Примерный вариант файла настроек:

```
[artix]
path = world
tPriceQR = 1
server_ip = 192.168.0.239
exchange_cat = //192.168.0.239/obmen/dict/
```

- При запуске функции «main()» происходит соединение с базой данных и создается объект для работы с ней. Работа с БД осуществляется в файле «db.py» Создаем экземпляр класса «workDb» для работы с базой данных передавая в качестве параметра при инициализации объект содержащий текущие настройки программы полученные из «ini» файла.

```
tData = db.workDb(rc)
```

Листинг 3 – Python example

При инициализации происходит подключение к БД и возвращается текущий курсор. В качестве параметров строка подключения содержит: IP-адрес сервера, имя БД, имя и пароль пользователя.

```
self.mydb = pymysql.connect(host=rc._sections.artix.server_ip,....."IPадрес – сервера"
                           database=rc._sections.artix.database,....."имя БД"
                           user=rc._sections.artix.user,....."имя пользователя"
                           passwd=rc._sections.artix.passwd)....."пароль"
self._mycursor = self.mydb.cursor() # cursor created
```

Листинг 4 – Python example

Так же создается объект для работы с Http запросами.

```
rec_con = m_request.req1C(rc)
```

Листинг 5 – Python example

- Получаем список смен которые были открыты с последней зафиксированной даты.

```
# Список открытых смен от последнего зафиксированного времени
l_workshift_open = tData.get_last_workshift_open()
```

Листинг 6 – Python example

Получение списка открытых смен с момента последнего обращения к БД

```
# Читаем из файла дату открытия последней отправленной в УНФ смены
l_date = self.load_last_date_open()

# Выполняем запрос с отбором по дате
# Текст запроса хранится в файле "diff_data.py"

# "SELECT shiftnum ,cashcode,CAST(time_beg AS char),shopcode FROM workshift WHERE
time_end IS NULL AND time_beg >%s '"

self._mycursor.execute(diff_data.qrSelect_workshift_open, [l_date])

# Результат выполнения запроса
l_workshift = self._mycursor.fetchall()
```

Листинг 7 – Python example

Если появились новые открытые смены, тогда формируем и отправляем Http запрос в 1С.

```
status_code = rec_con.post_workshift_open(l_workshift_open)
```

Листинг 8 – Http-запрос

С запросами работает файл «m\_request.py»

```
r = requests.post('http://' + self.mConfig._sections.one_C.server_ip + ':' + self.mConfig._sections.one_C.port + self.mConfig._sections.one_C.workshift_open,
                  data=None, json = l_workshift_open)
return (r.status_code)
```

Листинг 9 – Текст Http-запроса

Если код возврата был успешным (200), тогда меняем дату в файле, на дату открытия последней смены.

```
# Список открытых смен от последнего зафиксированного времени
l_workshift_open = tData.get_last_workshift_open()
# Если нечего отправлять, то не отправляем
if len(l_workshift_open) > 0:
    status_code = rec_con.post_workshift_open(l_workshift_open)
# Меняем дату в файле только в случае успешного результата работы 1С
if status_code == 200:
    tData.save_new_date_open()
else:
    logger.info('status_code_open - ' + str(status_code))
```

Листинг 10 – Python example

- Получаем список смен которые были закрыты с последней зафиксированной даты, если появились новые закрытые смены, тогда формируем и отправляем Http запрос в 1С. Если код возврата был успешным (200), тогда меняем дату в файле, на дату закрытия последней смены.

```
# Список закрытых смен от последнего зафиксированного времени
l_workshift = tData.get_last_workshift()
# Если нечего отправлять, то и не отправляем
if len(l_workshift) > 0:
    status_code = rec_con.post_workshift(l_workshift)
# Меняем дату в файле только в случае успешного результата работы 1С
if status_code == 200:
    tData.save_new_date()
else:
    logger.info('status_code_open - ' + str(status_code ))
```

Листинг 11 – Python example

- Завершаем работу программы.

## 3 Установка ПО

### 3.0.1. Установка OpenSSH

```
sudo apt install openssh-client openssh-server
```

### 3.0.2. Установка Nano

```
sudo apt install nano
```

# Список иллюстраций

1.1	«Формат Linux Crontab».	2
-----	-------------------------	---

---

## *Журнал изменений*

---

Версия	Дата	Автор(ы)	Изменения
0.01	27.01.23	РК	Начало работы
0.1	31.01.23	РК	Корректировка