ООО „КРЮГЕР ХАУС“

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

Конфигурация УНФ 3.0

Новосибирск, 2023 г.

Оглавление

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 >= 3.11
* Webmin (не обязательно) - панель для администрирования сервера
* OpenSSH (установка 4.0.1.)
* редактор Nano (установка 4.0.2.)

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:

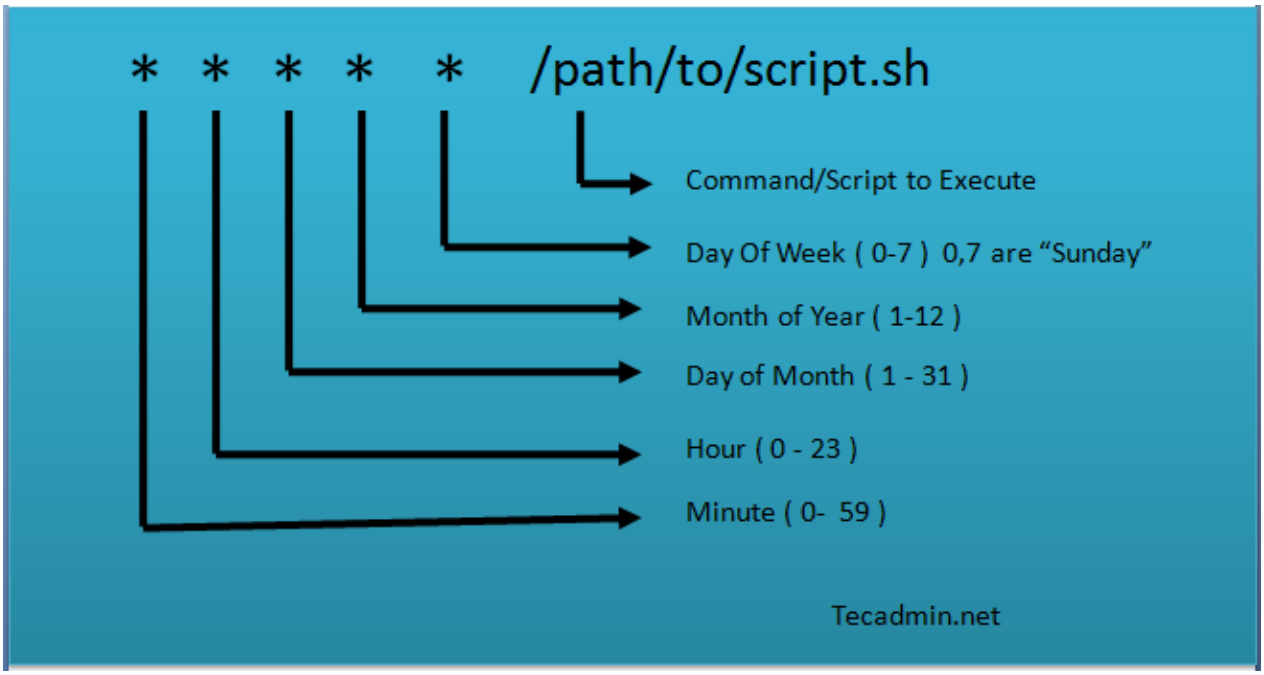

Рисунок 1.1 – «Формат Linux Crontab».

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

- /
  - Workshift_load
    - config — каталог с файлами настроек
      - config.ini — Файл содержит настройки программы.
    - log — каталог с файлами лога
      - py_log.log — файл лога
    - src — каталог с файлами программы
      - app_logger.py — работа с логированием
      - db.py — Работа с БД
      - diff_data.py — Файл содержит тексты запросов.
      - m_config.py — работа с конфигурацией
      - m_request.py — работа с Http-запросами
      - wsh_load.py — основной файл программы
      - last_date_open.txt — дата последней открытой смены ККМ
      - last_date.txt — дата последней закрытой смены ККМ
      - first.dat — Файл появляется после первого запуска программы если он отсутствует, то при запуске происходит инициализация файлов с датами смен ККМ.
  - requirements.txt — Файл содержит наименование и версии установленных модулей подключенных к программе, используется при начальной установке.

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 номерами смен равными «1». Таким образом при первом запуске по запросу с этими первыми номерами, в результат запроса попадут **все** кассовые смены имеющиеся в Artix, а номера 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('1')

    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('1')
```

Листинг 1 – Инициализация файлов с датами

- При запуске программы происходит чтение настроек. Если чтение удачно, 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 – Начало работы

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

```
; comment1
[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 предавая в качестве параметра при инициализации объект содержащий текущие настройки программы полученные из «ini» файла.

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

Листинг 3 – Создание объекта БД

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 – Соединение с БД

Так же создается объект для работы с Http запросами.

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

Листинг 5 – Объект запрос

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

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

Листинг 6 – Список открытых смен

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 – Смены из БД

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 + ':'..... IP адрес
                  + self.mConfig._sections.one_C.port.....Port
                  + self.mConfig._sections.one_C.workshift_open,..... Процедура
                   data=None, json = l_workshift_open)..... Данные смен ККМ
return (r.status_code)
```

Листинг 9 – Текст Http-запроса

В 1С данные отправляются в виде списка со значениями. Пример файла:

```
1 97, ..... Номер смены
2 'test', ..... Код кассы
3 '2023-02-06 13:35:38', ..... Дата открытия
4 'test' ..... Код магазина
```

Листинг 10 – Формат файла открытых смен

Если код возврата был успешным (**200**), тогда меняем номер смены, на номер последней отправленной в УНФ 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 1C
      if status_code == 200:
          tData.save_new_date_open()
      else:
          logger.info('status_code_open – ' + str(status_code ))
```

Листинг 11 – Обработка результата

- Получаем список смен которые были закрыты с последнего зафиксированного номера смены, если появились новые закрытые смены, тогда формируем и отправляем Http запрос в 1С.

  В 1С данные отправляются в виде списка со значениями. Пример файла:

```
96,..... ... Номер смены
'test', ..... Код магазина
'2023-02-06 13:33:40', ..... Дата открытия
'test', ..... Код кассы
'2023-02-03 10:02:25', ..... Дата закрытия
103, ..... Идентификационный номер смены
'_shop_test_56f60925', .... storeId
'_cash_test_e0146422', .... cashId
'3', ..... Код кассира
1, ..... Номер первого чека смены
3, ..... Номер последнего чека смены
'637.0000', ..... Сумма продажи
'637.0000', ..... Сумма выручки
'637.0000', ..... Сумма в денежном ящике
'2023-02-03 10:02:22', ..... Дата и время открытия первого чека в смене
'637.00', ..... Сумма продажи наличные()
'0.00', ..... Сумма продажи безналичные()
'0.00', ..... Сумма продажи прочие()
'637.00', ..... Сумма выручки наличные()
'0.00', ..... Сумма выручки безналичные()
'0.00', ..... Сумма возвратов
'0.00', ..... Сумма возвратов наличные()
'0.00', ..... Сумма возвратов безналичные()
3, ..... Количество чеков продажи
0 ..... Количество чеков возврата
```

Листинг 12 – Формат файла закрытых смен

Если код возврата был успешным (**200**), тогда меняем дату в файле, на номер последней отправленной в УНФ 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 1C
    if status_code == 200:
        tData.save_new_date()
    else:
        logger.info('status_code_open – ' + str(status_code ))
```

Листинг 13 – Закрытые смены

- Завершаем работу программы.

3 Описание работы УНФ

- В УНФ создан новый HTTP-сервис : «Workshift». Сервис может обрабатывать два метода;
  - «post_workshop» - для закрытых смен
  - «post_workshop_open» - для открытых смен

  При получении первого метода «post_workshop», в УНФ вызвывается процедура:

  ЗакрытьКассовыеСмены(ДанныеСменККМ);

  С передачей ей всех параметров смены.
  При получении второго метода «post_workshop_open», в УНФ вызвывается процедура:

  ОткрытьКассовыеСмены(ДанныеСменККМ);

  С передачей ей всех параметров смены.

4 Установка ПО

4.0.1. Установка OpenSSH `sudo apt install openssh-client openssh-server`

4.0.2. Установка Nano `sudo apt install nano`

5 Возможные проблемы

~~Ситуация, когда на кассах будет разное время, достаточно нескольких секунд. И тогда какая-то из открытых смен может не попасть в УНФ, если по времени кассы она открылась раньше уже зафиксированной даты, а по текущему времени позже.~~

Решено заменой даты на ID кассовой смены.

6 Приложение

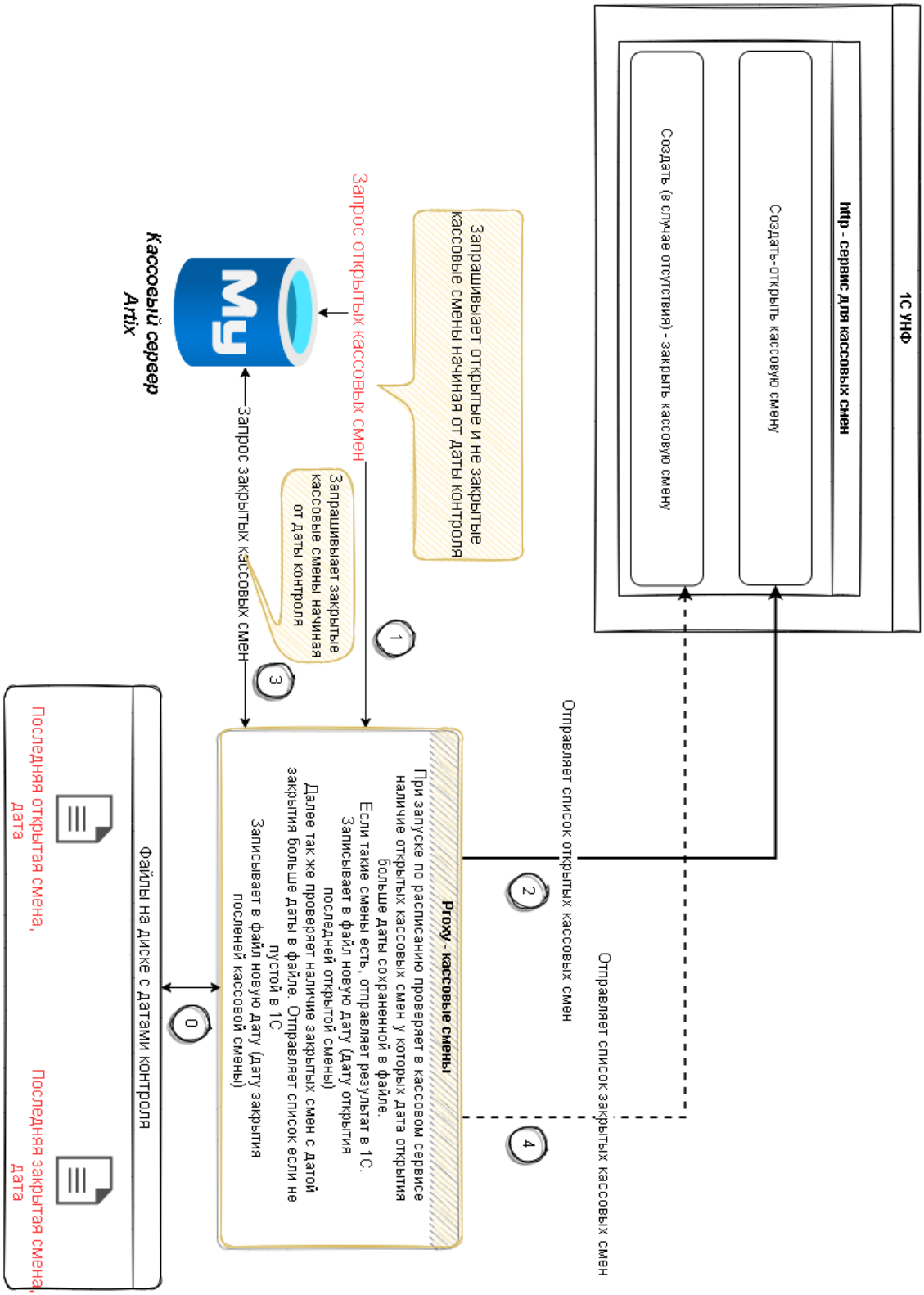

Рисунок 6.1 – «Общая схема»

Список иллюстраций

Листинги