Оглавление

[**1. Цели** 2](#_Toc493087588)

[**2. Структура** 3](#_Toc493087589)

[**3. Необходимые инструменты** 4](#_Toc493087590)

[**4. Расписание выполнения** 5](#_Toc493087591)

[**5. Результат выполнения** 6](#_Toc493087592)

[**6. Описание работы скриптов** 7](#_Toc493087593)

# **1. Цели**

Данное руководство описывает инструмент для автоматизации, главным и единственным назначением которого является упрощение процедуры составления некоторых отчетов группы мониторинга сети. Преследовалась цель минимизировать человеческий фактор в составлении и компоновке выгрузок с сервера мониторинга U2000, вследствие чего, очевидно, мы страхуемся от ряда ошибок.

Автоматизация производилась для выгрузок по следующим отчетам:

- суточный отчет KPI;

- VLR Бадашеву (суточный);

- VLR Шаркову (каждую пятницу);

- VLR Зенцову (каждую пятницу);

- отчет по утилизации транкгрупп (каждую пятницу);

- недельный отчет директорам (каждую пятницу);

- недельный отчет KPI в разрезе месяца (каждый понедельник).

# **2. Структура**

Назвать данный набор скриптов «системой автоматизации» не поворачивается язык – это именно набор скриптов и не более того. Все скрипты написаны на языке Python. Выполняются они на двух устройствах: на самом сервере U2000 и, кроме того, нужна рабочая станция для передачи файлов с сервера на общий ресурс (\\10.82.10.30\...). В качестве такой станции был выбран десктоп в кабинете мониторинга.

На сервере выполняются следующие скрипты:

- collect\_u2000\_export.py

- daily\_report\_data\_gathering.py

- traffic\_utilization\_data\_gathering.py

- vlr\_sharkov\_data\_gathering.py

- vlr\_zentsov\_data\_gathering.py

- weekly\_directors\_data\_gathering.py

- kpi\_weekly\_report\_data\_gathering.py

На рабочей станции выполняются скрипты:

- daily report.py

- traffic utilization.py

- vlr badashev.py

- vlr sharkov.py

- vlr zentsov.py

- weekly directors.py

- kpi weekly report.py

# **3. Необходимые инструменты**

Так как скрипты написаны на Python, нужна среда для их выполнения. На сервере U2000 этот вопрос по умолчанию решен – на большинстве серверов под управлением ОС семейства Linux (коим и является данный сервер – он работает под управлением SUSE Linux Enterprise Server 11 v3) Python идет из коробки. В данном случае уже стоял Python 2.6, отдельно нужно было поставить только библиотеку для работы с документами в формате .xlsx – openpyxl 2.4.0-a1 (это не последняя версия библиотеки, но последняя, работающая с 2.6 версией Python). Важный момент: установить ее удалось только из-под пользователя root (для ossuser не хватило прав), соответственно, скрипты, использующие эту библиотеку (а это все скрипты, выполняемые на сервере), должны быть также запущены из-под пользователя root.

Рабочая станция находится под управлением ОС Windows, соответственно, здесь интерпретатора Python по умолчанию нет. Он был установлен – версии 3.6.2, кроме того, здесь также была установлена библиотека openpyxl (версии 2.5.0-a3 – последняя на момент написания данного руководства). Также нужно было установить scp/sftp-клиент для забора скомпонованных данных с сервера – здесь выбор был остановлен на WinSCP.

# **4. Расписание выполнения**

Содержимое файла crontab на сервере, выполняемое от пользователя root и касающееся только рассматриваемой в рамках руководства предметной области:

30 03 \* \* \* cd /opt/oss/server/var/fileint/pm/NOC\_reports/Daily\_report && /usr/bin/python daily\_report\_data\_gathering.py

30 06 \* \* \* cd /opt/oss/server/var/fileint/pm/NOC\_reports/KPI\_weekly\_report/ && /usr/bin/python kpi\_weekly\_report\_data\_gathering.py

30 07 \* \* \* cd /opt/oss/server/var/fileint/pm/NOC\_reports/Traffic\_utilization && /usr/bin/python traffic\_utilization\_data\_gathering.py

00 08 \* \* \* cd /opt/oss/server/var/fileint/pm/NOC\_reports/VLR\_Sharkov && /usr/bin/python vlr\_sharkov\_data\_gathering.py

30 08 \* \* \* cd /opt/oss/server/var/fileint/pm/NOC\_reports/VLR\_Zentsov && /usr/bin/python vlr\_zentsov\_data\_gathering.py

00 09 \* \* \* cd /opt/oss/server/var/fileint/pm/NOC\_reports/Weekly\_directors && /usr/bin/python weekly\_directors\_data\_gathering.py

45 07 \* \* 5 cd /opt/oss/server/var/fileint/pm/NOC\_reports/Traffic\_utilization && chmod 777 \*.xlsx

15 08 \* \* 5 cd /opt/oss/server/var/fileint/pm/NOC\_reports/VLR\_Sharkov && chmod 777 \*.xlsx

45 08 \* \* 5 cd /opt/oss/server/var/fileint/pm/NOC\_reports/VLR\_Zentsov && chmod 777 \*.xlsx

15 09 \* \* 5 cd /opt/oss/server/var/fileint/pm/NOC\_reports/Weekly\_directors && chmod 777 \*.xlsx

Здесь нужно отметить два момента. Во-первых, время здесь указано со смещением в 5 часов (то есть, если, например, здесь написано, что скрипт выполняется в 08:00 – на самом деле он выполняется в 03:00). С чем это смещение связано – я не выяснял за ненадобностью, просто нужно это иметь в виду. При этом команда date корректно отображает время.

Второй момент – установка прав доступа к xlsx файлам в 777 (четыре последние строки). Это нужно для того, чтобы сторонний scp-клиент (WinSCP в данном случае) смог забрать эти файлы с сервера – иначе возникнет ошибка. Файлы для дневного отчета (Daily report.xlsx) и для недельного отчета в разрезе месяца (KPI weekly report.xlsx) не удаляются и просто перезаписываются каждый день, для них уже установлены права 777, поэтому для них отдельно устанавливать уже не нужно. Но можно выставить вручную один раз, если что-то сбилось.

На рабочей станции скрипты выполняются по такому расписанию (созданы соответствующие задачи в планировщике):

В 23:00 каждый день – daily report.py

В 23:30 каждый день – vlr badashev.py

В 02:30 по понедельникам – kpi weekly report.py

В 03:00 по пятницам – traffic utilization.py

В 03:30 по пятницам – vlr sharkov.py

В 04:00 по пятницам – vlr zentsov.py

В 04:30 по пятницам – weekly directors.py

Здесь комментарии излишни: каждая задача на рабочей станции выполняется спустя некоторое время после того, как отработает соответствующий ей скрипт на сервере.

# **5. Результат выполнения**

После выполнения всех скриптов работа над соответствующими отчетами сводится к такому состоянию:

1. VLR Бадашеву – файлы с VLR с расширением .txt за текущий день уже присутствуют в папке на общем ресурсе (\\10.82.10.30\Share\ТД\Мониторинг\Отчёты\VLR\_Бадашеву\мм.гггг\);
2. VLR Зенцову – в папке на общем ресурсе (\\10.82.10.30\Share\ТД\Мониторинг\Отчёты\Недельный отчёт VLR Зенцову - 9ч. пятница\) каждую пятницу формируется уже готовый файл с именем VLR дд.мм.гггг.xlsx – в соответствии с текущей датой;
3. VLR Шаркову – в папке на общем ресурсе (\\10.82.10.30\Share\ТД\Мониторинг\Отчёты\Недельный отчёт VLR Шаркову - 9ч. пятница\) формируется файл VLR Subscribers дд.мм.гггг.xlsx, агрегированные данные из которого можно копировать в исходный файл KPI\_report\_VLR.xlsx;
4. Недельный отчет по утилизации транкгрупп – в папке на общем ресурсе (\\10.82.10.30\Share \ТД\Мониторинг\Отчёты\Недельный отчёт ТГ Гончаруку – пятница\) данные за неделю уже будут внесены в файлы Report Traffic Trunk Groups ISUP.xlsx и Report Traffic Trunk Groups SIP.xlsx;
5. Недельный отчет директорам – в папке на общем ресурсе (\\10.82.10.30\Share\ТД\Мониторинг\Отчёты\Недельный отчёт директорам-к 11ч. пятницы\Report\) данные по каждому пункту из отчета уже скопированы в файлы с именами 11.xlsx в соответствующих папках. Достаточно только обновить шаблон отчета;
6. Суточный отчет – после выполнения скрипта в папке на общем ресурсе (\\10.82.10.30\Share\ТД\Мониторинг\Отчёты\Суточный отчет\ггггммм) уже сформирована папка «дд месяца гггг\», где дата – сегодняшнее число, с тремя файлами: два из них – из отчета за вчерашний день (презентация и excel-файл), третий – файл с именем Daily report.xlsx с готовой актуальной выгрузкой по всем пунктам. То есть, достаточно просто перенести данные из файла Daily report.xlsx в файл KPI\_report\_5.xlsx, обновить презентацию и удалить файл Daily report.xlsx (последнее, в принципе, необязательно);
7. Недельный отчет KPI в разрезе месяца – аналогично суточному отчету. Также создается папка по имени текущей даты с файлами из прошлого отчета (за прошлое воскресенье/понедельник), плюс добавляется файл KPI weekly report.xlsx. Данные из него достаточно перенести в файл KPI\_report.xlsx и обновить презентацию.

Такое деление, что для некоторых отчетов готовые данные уже упакованы в нужные файлы, а для других они готовы, но их нужно скопировать в конечные, обусловлено следующим: к сожалению, модуль openpyxl при открытии xlsx файла удаляет все графики в нем. Таким образом, для тех отчетов, где в файлах есть графики (например, суточный отчет или VLR Шаркову) перезапись в текущий файл не ведется, а по соседству формируется новый, из которого уже можно вручную перенести данные. Более удобной альтернативы модулю openpyxl я не нашел, поэтому пришлось пойти на такой компромисс.

# **6. Описание работы скриптов**

В данном разделе описание работы скриптов будет рассмотрено с большой долей обобщения, иными словами, не будет поясняться, как работает та или иная конкретная строчка в коде. Хоть в нем и есть много нюансов и подгонок под конкретные входные данные, тем не менее, такое подробное описание будет занимать не одну и не две страницы.

Весь функционал можно условно разбить на три части:

- скрипт collect\_u2000\_export.py, реализующий основную логику компоновки данных и использующийся в скриптах из следующей категории;

- скрипты сбора данных для каждого конкретного отчета (это скрипты, выполняемые на сервере и имеющие суффикс \_data\_gathering.py в названии);

- скрипты, выполняющиеся на рабочей станции.

Так как скрипты из одной группы реализуют примерно одну и ту же логику, рассматривать их так же будем по группам.

Первая группа:

Сюда входит только один скрипт collect\_u2000\_export.py и он не выполняется напрямую. Из него на экспорт идет главная функция, а остальные являются внутренними и вызываются из главной. Краткое описание функций:

collect\_data() – главная функция, идущая на экспорт в скрипты из второй группы. Вызывает в себе четыре внутренние функции: \_compose\_files\_list, \_open\_workbook, \_append\_to\_workbook и \_set\_numeric\_types.

\_compose\_files\_list() – возвращает список файлов, из которых были собраны данные. Обычно это файлы за вчерашний день с 00:00 до 00:00 текущего дня, исключение – суточный отчет (там данные с 21:00 вчерашнего дня по 21:00 текущего).

\_open\_workbook() – открывает документ .xlsx с заданным именем, если такого не существует – создает его. Также для суточного отчета и недельного отчета KPI «удаляются» старые данные (за неделю назад и месяц назад соответственно). Удаляются в кавычках, потому что на самом деле они не стираются напрямую, а просто данные в документе сдвигаются на определенное количество строк вверх, а свежие данные всегда дозаписываются в конец.

\_append\_to\_workbook() – запись новых данных в документ из предыдущей функции. Данные берутся из файлов из списка, сгенерированного функцией \_compose\_files\_list и дозаписываются в конец книги, открытой функцией \_open\_workbook.

\_set\_numeric\_types() – к сожалению, openpyxl сохраняет все данные в текстовом формате. Это не всегда удобно при копировании и дальнейшей обработке в Excel, поэтому для данных в числовом формате лучше жестко задать тип прямо в документе. Это и делает данная функция.

Вторая группа:

Скрипты из этой группы очень коротки. Они импортируют в себе главную функцию из collect\_u2000\_export.py – collect\_data() и запускают ее с разными параметрами в зависимости от того, какой отчет мы составляем. Например, для отчетов по VLR нам нужен только сабсет, связанный с данными по количеству абонентов, для суточного отчета – множество сабсетов для разных KPI и т.п.

Третья группа:

Скрипты, выполняемые на рабочей станции. По большому счету, все они имеют одинаковую структуру – это набор shell-команд, вызываемых из Python с помощью модуля subprocess. Как правило, первая команда – запуск WinSCP для забора файлов с сервера на локальный диск, затем идет сброс их на общее хранилище в соответствующие для каждого отчета папки. Для некоторых отчетов (где нет графиков) данные сразу записываются в файлы на общие ресурсы. По окончанию выполнения этих скриптов получаем состояние, описанное в пункте 5 данного руководства.