Платформа управления пентестами и организации команд

Руководство администратора

Модуль Apiary

Краткая информация о Apiary

Модуль **Apiary** платформы **ПЭВМ «Платформа управления пентестами и организации команд»** - это рабочее пространство, предназначенное для сбора данных и управления аудитами безопасности. **Apiary** помогает координировать действия не только команд по аудиту безопасности, но и членов собственной команды.

Модуль **Apiary** тесно связан с модулем **Hive**. Вся информация в **Apiary** поступает из проектов **Hive**. Таким образом, в **Apiary** можно наглядно посмотреть общее количество ишей, где они были обнаружены, определить опасность этих ишей, а также назначить ответственного за их устранение.

Для работы с большим объемом информации в **Apiary** доступно создание и кастомизация фильтров. Используйте фильтры для сравнения данных, просмотра информации об ишах, а также для просмотра статуса устранения ишей.

Привлекайте к тестированию столько команд, сколько необходимо для полноценного аудита. С помощью **Apiary** вы можете общаться с командами по аудиту на всех этапах тестирования, а также сравнивать результаты и оценивать эффективность разных команд.

Используйте **Apiary**, чтобы делиться результатами аудита с коллегами и распределять все обнаруженные иши прямо в **Apiary** в виде задач.

Настоящее руководство составлено для ознакомления пользователя с административными функциями и настройками **Apiary**. Пользуясь данным руководством, пользователь может самостоятельно установить, обновить, создать резервную копию, и выгрузить логи системы в случае возникновения неполадок.

Руководство предназначено для системных администраторов.

В основной части документа приведены сведения о назначении, возможностях, настройках и об управлении системой.

Для удобства работы руководство поделено на разделы, главы и параграфы.

Установка и обслуживание

Apiary предполагает развёртывание в инфраструктуре заказчика на отдельной виртуальной или физической машине. Инсталляция производится при помощи дистрибутивного пакета (бандла), представляющего собой самораспаковывающийся архив с инсталлятором.

В этой главе описаны предварительные требования к машине, процесс инсталляции и обновления, а так же информация по обслуживанию **Apiary**.

Требования к оборудованию

Аппаратные требования

Ниже указаны системные требования:

Описание	Минимальные требования	Рекомендованные требования
RAM (ГиБ)	4	12
CPU (ядер)	2	4
Скорость диска (IOps)	100	300
Место на диске (ГиБ)	100	200

Поддерживаемые операционные системы

Перед установкой подготовьте виртуальную или физическую машину с одной из поддерживаемых операционных систем:

- RHEL 8
- Ubuntu 22.04
- Ubuntu 24.04
- Debian 11
- Debian 12
- RedOS 7.3
- RedOS 8.0
- Astra Linux Орёл 1.7

- другие ОС, совместимые с указанными выше.

Дополнительное общее программное обеспечение

Некоторые дистрибутивы Linux в минимальной установке могут не включать следующие утилиты: `curl`, `find`, `groupadd`, `tar`, `gzip`, `useradd`, `xargs`. Однако эти утилиты используются в **Apiary**, и мы рекомендуем вам установить их.

Система контейнеризации

Установите docker engine, включая docker compose plugin на машину.

Несколько важных примечаний:

1. `docker from snap` на данный момент **не поддерживается**.
2. docker-compose standalone поддерживается, но не рекомендуется.
3. Несмотря на то, что разработчики `podman` заявляют о совместимости с `docker` по API и CLI **Apiary** не работает с `podman`.
4. Во многих дистрибутивах Linux Вы можете установить `docker` из пакетной базы данного дистрибутива вместо установки с docker.com. Однако, в этом случае, могут потребоваться дополнительные шаги, о которых написано ниже.

Известные сложности при использовании `docker` и `docker compose plugin` из репозиториев ОС, вместо docker.com:

- В репозиториях некоторых ОС есть только docker-compose 1.x который не поддерживается **Apiary**. Для установки `docker compose plugin` воспользуйтесь инструкцией со страницы Install the plugin manually, но обратите внимание, на пометки `for all users`. Если плагин будет установлен только текущего пользователя, даже если это будет `root`, запустить **Apiary** корректно невозможно.
- В репозиториях некоторых ОС `docker` после установки не стартуется автоматически или не создаёт `socket`. В этих случаях его нужно запустить вручную.

Установка

Загрузка установочного файла

Скачайте установочный файл на целевую машину, то есть, на машину, на которую вы планируете установить **Apiary**. Есть два варианта установочного файла:

- *Онлайн*. Размер установочного файла чуть больше 100 килобайт. В процессе установки все необходимые для работы образы контейнеров

будут скачаны на вашу машину. Для инсталляции на целевой машине требуется доступ в интернет. URL: https://hexway.io/download/apiary/latest.

- *Оффлайн*. Размер установочного файла несколько сотен мегабайт (**Осторожно трафик!**). Однако, все необходимые для работы образы контейнеров упакованы внутрь инсталлятора. Для инсталляции на целевой машине не требуется доступ в интернет. URL: https://hexway.io/download/apiary-offline/latest.

Для того что бы скачать инсталлятор на машину вы можете воспользоваться доступным Вам инструментом:

- `curl` - можно скачать прямо с целевой машины. Рекомендуем добавить опции `--location --remote-header-name --remote-name`. Например, так: `shell curl --location --remote-header-name --remote-name https://hexway.io/download/apiary/latest`

- `wget` - так же можно скачать прямо с целевой машины. Рекомендуем добавить опцию `--content-disposition`.
  Например, так: `shell wget --content-disposition https://hexway.io/download/apiary/latest`

- Или скачать по указанной выше ссылке локально на ваш десктоп и залить на целевую машину любыми доступными вам способами (`scp`, `ftp`, `smb` и т.д.).

Установка

Для запуска установки выполните команду:

```
sudo bash ./install_hw_fh_en-US_2025.9.1.run
```

или для offline инсталлятора:

```
sudo bash ./install_offline_hw_fh_en-US_2025.9.1.run
```

По завершению установки **Apiary** автоматически запустится и вы увидите следующее сообщение:

```
hexway Apiary ready to use.
UI accessible on http://0.0.0.0 or https://0.0.0.0
Login as root@ro.ot password: <password>
```

⚠ При установке **Apiary** без использования дополнительных параметров, пароль пользователя `root@ro.ot` будет сгенерирован автоматически.

Теперь вы можете открыть **Apiary** в браузере, например: `http://<ip-server>/` или `http://<your-domain-name>/`, если у вас есть доменное имя указывающее на этот IP адрес.

Что может быть важно:

- Что-то пошло не так? Посмотрите главу ЧаВо, возможно, что ответ на ваш вопрос уже есть.
- У нашего инсталлятора есть ряд дополнительных параметров. Возможно, что эти параметры могут вам пригодиться.
- Здесь вы можете узнать как останавливать, запускать, обновлять и конфигурировать **Apiary**.
- Скорее всего, вы хотите настроить доступ к **Apiary** с использованием `https`. Здесь вы можете узнать как это можно сделать.
- Возможно, что вы не хотите запускать на вашей машине что-то скачанное из интернета, с правами `root` не изучив это. здесь вы можете познакомиться с тем, как можно распаковать и изучить содержимое инсталлятора.

Пароли по умолчанию

После первого логина вам будет предложено сменить пароль суперпользователя `root@ro.ot`. Настоятельно рекомендуем вам это сделать и сохранить новый пароль в надёжном месте.

Пароль суперпользователя `root@ro.ot`, сгенерированный во время установки Apiary сохраняется в файл `/opt/hw-fh/config/local.ini`. Если вы стёрли содержимое терминала, не сохранив пароль, вы можете его посмотреть при помощи команды:

```
/opt/hw-fh/bin/show-info
```

⚠ Как только процесс начальной инсталляции завершён, пароль суперпользователя, из файла `/opt/hw-fh/config/local.ini` больше не используется. А если вы, следуя нашим рекомендациям, сменили его после первого логина, то пароль, хранящийся в файле `/opt/hw-fh/config/local.ini` не имеет значения - он устарел.

Пути к файлам

Все файлы хранятся в директориях, перечисленных ниже:

- `/opt/hw-fh/` - базовая директория для хранения данных, конфигурации продукта, управляющих скриптов и прочих вспомогательных файлов;
- `/opt/hw-fh/data` - в этой директории хранятся все изменяемые файлы, такие как:
  - данные встроенных в продукт СУБД (`postgresql`, `redis`, etc);
  - файлы, проектов и ишей;
  - кеш обратного прокси сервера reverse proxy server (`nginx`);

- `/opt/hw-fh/config` - в этой директории хранятся все конфигурации **Apiary**:
  - конфигурации, предоставленные в дистрибутиве;
  - конфигурации, сформированные в процессе установки;
  - пользовательские конфигурации - файл `/opt/hw-fh/config/users.ini`. Пример заполнения файла `users.ini` можно посмотреть в файле `/opt/hw-fh/config/user-template.ini`;
- `/opt/hw-fh/compose`: в этой директории хранится файл docker-compose, сформированный конфигуратором продукта из конфигурации, хранящейся в `/opt/hw-fh/config`;
- `/opt/hw-fh/systemd`: в этой директории хранятся файлы шаблонов для `systemd` модуля и используемый в настоящее время systemd модуль.

Дополнительные параметры установки

Бывает, что нельзя просто так взять и установить **Apiary** из бандла на целевую машину. Может потребоваться:

- изучить содержимое, что бы убедиться, что здесь нет ничего ломающего;
- задать дополнительные параметры ещё до первой установки;
- получить дополнительную информацию о процессе инсталляции в случае каких-то проблем при установке.

В этом параграфе описано как это можно сделать.

Что собой представляет бандл

Бандл представляет собой `shell` скрипт и архив в формате `.tar.gz` в одном файле. Он создан при помощи makeself и его задача просто распаковать во временную директорию содержимое архива и запустить скрипт инсталляции (`embedded script` в терминологии `makeself`).

Вы можете просмотреть текст скрипта `makeself` в любом текстовом редакторе, что бы убедиться, что в нём нет ничего подозрительного.

Ещё в нём есть опции. Вы можете посмотреть все опции, например, так:

```
sudo bash ./install_hw_fh_en-US_2025.9.1.run --help
```

Наиболее интересные, с практической точки зрения опции:

- `--noexec`: не запускать скрипт инсталляции после распаковки.
- `--target`: распаковать содержимое в указанную директорию вместо временной.
- `--`: все последующие аргументы передать в скрипт инсталляции.

Извлечение файлов инсталляции

Иногда требуется извлечь файлы из бандла. Это может потребоваться для того, что бы изучить его содержимое или для того что бы провести диагностику ошибки в случае, если инсталляция ломается по непонятной причине.

Пользуясь опциями из предыдущего параграфа, вы можете просто распаковать весь архив с Apiary таким образом:

```
sudo bash ./install_hw_fh_en-US_2025.9.1.run --noexec --target 2025.9.1
```

После выполнения этой команды будет создана директория `2025.9.1` в которую будет расковано всё содержимое архива.

Инсталляция из распакованного бандла

Если вы распаковали бандл, как описано в предыдущем параграфе, то вы можете перейти в директорию, в которую распаковали бандл, и запустить инсталлятор из неё:

```
sudo ./installer
```

Или так: если требуется диагностировать работу инсталлятора:

```
sudo bash -x ./installer
```

Содержимое архива бандла

Рассмотрим содержимое бандла.

имя файла или директории	назначение	примечания
`ascii-logo.txt`	Логотип, которое отображается в консоли при установке	
`bin/`	Директория с bash скриптами для распаковки в `/opt/hw-fh/bin`	Копируются в `/opt/hw-fh/bin`
`config/defaults.ini`	Глобальная конфигурация версии приложения	Копируется в `/opt/hw-fh/config/`
`config/user-template.ini`	Пример всех настроек приложения, доступных пользователю	Копируется в `/opt/hw-fh/config/`
`images_list.txt`	Список образов контейнеров, входящих в состав Apiary	
`installer`	Скрипт инсталляции	

имя файла или директории	назначение	примечания
`installer.sh`	Дополнительные, используемые скриптом инсталляции	
`kb/`	Шаблоны отчётов	Копируется в `/opt/hw-fh/kb`
`dpssl/`, `fssl/`, `qssl/`	Дефолтные директории для SSL сертификатов и приватных ключей для `HTTPS`	
`systemd/`	Директория с шаблонами systemd модулей	Копируется в `/opt/hw-fh/systemd`
`*.tgz`	Образы контейнеров	Только для offline бандла

Вы можете изучить содержимое скриптов инсталляции и логику, по которой они работают.

Опции скрипта инсталляции

Но опции есть не только у `makeself`, но и у скрипта инсталляции. Что бы увидеть их вы можете сделать:

```
sudo bash ./install_hw_fh_en-US_2025.9.1.run -- --help
```

Наиболее интересные опции:

- `--nostart`: не стартовать автоматически Apiary после установки.
- `--noconfig`: не делать автоматическую переконфигурацию Apiary в процессе инсталляции ('nostart' подразумевается).

Экспериментальные функции (не рекомендуем для производственного окружения):

- `--foreign-docker-host`: указать docker host для rootless докера.
- `--foreign-docker-user`: указать пользователя, под которым работает rootless docker.

Пример: вы хотите произвести установку, но не запускать и не конфигурировать **Apiary** для того что бы задать свой собственный начальный пароль (или произвести другие тонкие настройки):

```
sudo bash ./install_hw_fh_en-US_2025.9.1.run -- --nostart --noconfig
```

После этого вы можете открыть файл `/opt/hw-fh/config/users.ini` на редактирование, и в секции `[main]` добавьте, например, такую строку:

```
[main]
b.root.password = <new_password>
```

Чтобы изменения вступили в силу, выполните следующую команду:

```
sudo /opt/hw-fh/bin/reconfig
```

Apiary автоматически запустится, и у пользователя root будет установлен указанный дефолтный пароль.

Подробнее про настройки, которые можно выполнить в файле `/opt/hw-fh/config/users.ini` смотрите здесь.

Обслуживание

Запуск и остановка

После установки Apiary работает как `systemd` сервис. Поэтому, управлять им можно так же, как обычным `systemd` сервисом:

- Статус `systemd` сервиса Apiary: `shell systemctl status hw-fh`
- Останов Apiary: `shell systemctl stop hw-fh`
- Старт Apiary: `shell systemctl start hw-fh`
- Рестарт Apiary: `shell systemctl restart hw-fh`

После рестарта машины сервис так же автоматически запускается как обычный `systemd` сервис, после успешного старта `docker`.

⚠ Обратите внимание, что в случае изменения конфигурации простого рестарта недостаточно! Для того, что бы изменения вступили в силу требуется реконфигурация.

Установка обновлений

Чтобы обновить **Apiary**:

- Скачайте установочный файл на целевую машину аналогично тому, как делали это в процессе инсталляции.
- Запустите установку, так же как вы делали в процессе инсталляции.

Во время установки обновлений, сперва будет сохранена резервная копия **Apiary** (это может занимать значительное времени). Подробнее про резервные копии Apiary можете прочитать здесь. После этого будет установлено обновление.

После установки обновлений - **Apiary** автоматически запустится. - Соединения между проектами (**Hive** и **Apiary**) должно автоматически восстановиться, однако для этого может потребоваться время.

Если чего-то из этого не произошло, смотрите главу "Диагностика неполадок".

Реконфигурация

Для изменения конфигурации **Apiary** вы можете изменять файл `/opt/hw-fh/config/user.ini`.

Для применения изменений, которые вы сделали в файле `user.ini` нужно выполнить команду:

```
/opt/hw-fh/bin/reconfig
```

⚠ Обратите внимание:

- Есть несколько свойств, изменение которых не требует выполнения `reconfig`. В этих случаях в документации будет помечено особым образом. Если не сказано ничего другого, то `reconfig` обязателен.
- Файлы `defaults.ini` и `local.ini` в директории `/opt/hw-fh/config/` не предназначены для изменения пользователем. Файл `defaults.ini` будет обновляться при каждом обновлении **Apiary**. Файл `local.ini` содержит техническую информацию и может обновляться при каждой реконфигурации **Apiary**.
- Список всех свойств, которые можно задавать в `user.ini` вы можете найти в файле `user-template.ini` в той же директории. Подробнее о некоторых наиболее важных из них описано в различных параграфах в главе Настройки.

Возможно, наиболее важные настройки: - настройка портов; - настройка https для доступа **Apiary**; - добавление корневого сертификата для TLS доступа к сторонним системам при настройке интеграции.

Удаление

Если, по какой-то причине, вам требуется удалить **Apiary** с машины, вы можете сделать следующее:

- Остановите **Apiary**: `shell systemctl stop hw-fh`
- Удалите `systemd` сервис: `shell systemctl disable hw-fh`
- Удалите директорию с **Apiary**: `shell rm -rf /opt/hw-fh`
- Удалите директорию с резервными копиями: `shell rm -rf /opt/hw-fh_backup`

Использование rootless docker

Вы можете установить Apiary так что бы использовался `docker rootless`. Для этого вам нужно:

- Запустить `docker` в режиме `rootless`.
- Настроить его для нормальной работы с **Apiary**.
- Установите или переустановите **Apiary** с rootless docker.

Запуск docker в режиме rootless

Для того, что бы запустить docker в режиме rootless можно воспользоваться инструкцией на сайте разработчика `docker`. Подробную инструкцию можно найти по ссылке: https://docs.docker.com/engine/security/rootless/. Инструкция в данном параграфе, основана на этой инструкции и дополнена некоторыми пояснениями и уточнениями.

1. Выберите или создайте отдельного пользователя, под которым будет запущен docker. Для этого нельзя использовать пользователя `docker`. Пользователем `hw-fh-srv` воспользоваться также нельзя. В нашем примере мы назвали пользователя `test-user`, но рекомендуем использовать именование, принятое в вашей организации.

   ```
   useradd test-user && passwd test-user
   ```

2. Проверьте, что установлен пакет `rootlesskit`:

   ```
   rootlesskit --version
   ```

3. Для корректной работы Вам потребуется `iptables`. Проверьте, что он у вас уже уставлен, если нет, то необходимо его установить. Так же может потребоваться загрузка модуля ядра `iptables`. Например, для операционных систем семейства RHEL и других совместимых:

   ```
   dnf install -y iptables modprobe ip_tables
   ```

4. Из-под пользователя `root` остановите и выгрузите сервис hw-fh, если он был запущен ранее:

   ```
   systemctl stop hw-hw systemctl disable hw-hw
   ```

5. Очистите docker от хранения лишних данных на машине:

   ```
   docker system prune --all --volumes --force
   ```

6. Выгрузите `rootful` сервис `docker` из-под пользователя `root`:

   ```
   systemctl disable --now docker.service docker.socket
   ```

7. Установите `rootless docker` из-под нового пользователя. Для этого зайдите в систему под новым пользователем (`test-user` в данном примере). Примечание: Использовать команду `su` не получится,

необходимо войти через SSH или используя консоль виртуальной машины, чтобы зайти под этим пользователем. Команда для установки: `dockerd-rootless-setuptool.sh install`

8. Обратите внимание, на предложение добавить `docker socket` в переменную `DOCKER_HOST` в `~/.bashrc`. Это может быть полезно в дальнейшем. Кроме того, этот `docker socket` потребуется нам в дальнейшем - сохраните его себе. Примечание: здесь `1001` это `uid` вашего пользователя для запуска `docker`, у вас может быть другое значение. `echo "export DOCKER_HOST=unix:///run/user/1001/docker.sock" >> ~/.bashrc`

9. Активируйте rootless docker сервис из-под его пользователя (`test-user` в нашем примере): Обратите внимание: Для всех операций с сервисом `docker` при помощи команд `systemctl` и `journalctl` теперь нужно использовать опцию `--user`:

   ```
   systemctl --user enable docker
   ```

10. Запустите сервис `docker`: `systemctl --user start docker`

Дополнительная настройка rootless docker

1. Дополнительные настройки docker можно будет добавить в файле конфигурации `~/.config/docker/daemon.json`. Вы можете в этом файле указывать, например, диапазоны ip адресов для создаваемых контейнеров, настройки зеркалирования для docker registry и тому подобное.
   Создайте файл, как в этом примере, что бы добавить эти настройки в будущем:

   ```
   mkdir ~/.config/docker touch daemon.json
   ```

2. После выполнения шагов и предыдущего параграфа сервис `docker` запускается при первом логине их под пользователя, под которым он установлен (`test-user` в нашем примере).
   Чтобы сервис docker запускался не только при входе в систему под этим пользователем, но и сразу при старте машины, необходимо сделать (под пользователем root):

   ```
   loginctl enable-linger test-user
   ```

3. Чтобы можно было смотреть логи docker под этим пользователем сделайте следующие шаги (требуются права `root`):

   - В файле `/etc/systemd/journald.conf` укажите `Storage` в значении `persistent`, если вы это не делали ранее.

     ```
     Storage=persistent
     ```

   - Перезапустите `journald`:

     ```
     systemctl restart systemd-journald
     ```

- Теперь, чтобы посмотреть логи `docker`, выполните команду из-под пользователя, под которым запущен `rootless docker`:

  ```
  journalctl --user docker
  ```

4. Если вы не используете сторонний обратный прокси, и чтобы бы, при этом, **Apiary** был доступен на порту 80 и/или 443, необходимо выполнить команду (требуются права root):

   ```
   setcap cap_net_bind_service=ep /home/test-user/bin/
   rootlesskit
   ```

5. Для применения настроек `docker` нужно перезапустить `docker` сервис из-под пользователя, под которым запущен (`test-user` в нашем примере):

   ```
   docker.systemctl --user restart docker
   ```

6. В зависимости от вашего дистрибутива Linux и его настроек по умолчанию, возможно вам потребуется открыть входящий порт в `firewall` для каждого из нужных вам портов 80, 443, 5671, 5672. Например, для порта 443:

   ```
   firewall-cmd --zone=public --permanent --add-port 443/tcp
   ```

7. Для применения настроек firewall перезапустите его: `firewall-cmd --reload`

Установка с rootless docker

После того как `docker` установлен и настроен в режиме `rootless`, установите **Apiary** с использованием rootless docker.

Для установки необходимо знать:

- имя пользователя, под которым запущен `docker`;
- `docker socket`, который ранее вам было предложено сохранить в переменную `DOCKER_HOST`.

Укажите эти значения в качестве параметров установочного бандла:

```
sudo bash install_hw_fh_en-US_2025.9.1.run -- \
--foreign-docker-host unix:///run/user/1001/docker.sock \
--foreign-docker-user test-user
```

Обратите внимание, что сперва требуется указать два минуса, и затем опции установщика:

- `--foreign-docker-host` – имя сокета docker;
- `--foreign-docker-user` – имя пользователя, под которым запущен docker.

После окончания работы инсталлятора вы должны получить **Apiary** работающим с **rootless docker**.

Резервное копирование

Как делается резервное копирование

Резервные копии сохраняется:

- при каждом обновлении Apiary автоматически;
- вручную по команде пользователя (или из другого скрипта, например, `cron`).

Функциональность резервного копирования реализована в форме двух сервисных утилит для создания и восстановления резервной копии. Для выполнения резервного копирования доступно 2 опции:

- с полной остановкой всех сервисов ("холодная" копия).
- с частичной остановкой сервисов ("тёплая" копия). В этом случае время останова и старта приложения может быть меньше, чем, при выполнении "холодной" резервной копии.

В обоих случаях, при выполнении резервного копирования производится копирование всех сырых данных с диска.

Настройки резервного копирования

Резервные копии сохраняются в папку `/opt/hw-fh_backup`.

По умолчанию хранятся только последние 3 копии. Это значение можно изменить при помощи опции `product.backup.depth` в `user.ini` для этой опции можно не выполнять reconfig) или задать в качестве аргумента командной строки в случае создания ручной резервной копии.

Старые резервные копии будут удаляться, если лимит хранимых резервных копий превышен.

Создание резервной копии

Чтобы создать резервную копию **Apiary**, выполните с правами администратора:

```
/opt/hw-fh/bin/full-backup
```

Дополнительные опции:

- `-f`, `--force`: создание резервной копии без подтверждающего вопроса;
- `-w`, `--warm`: создаёт "тёплую" резервную копию: останавливаются, все службы за исключением служб хранения.
- `-h`, `--help`: краткая информация об опциях резервного копирования;

- `backup_depth` - количество хранимых резервных копий (целое число). Если не указано, то берется значение из параметра `product.backup.depth` в файле `user.ini`. Если и там не задано, то 3.

Обратите внимание, что этот скрипт можно запускать не только вручную, но и, например, из `cron`. Мы рекомендуем делать резервные копии при помощи `cron` и `/opt/hw-fh/bin/full-backup` регулярно.

Восстановление данных из резервной копии

Для восстановления из резервной копии выполните команду:

```
/opt/hw-fh/bin/backup-restore
```

Дополнительные опции:

- `-h`, `--help`: краткая информация об опциях резервного копирования.
- `-f`, `--file`: явное указание пути до файла резервной копии. Если не указано, то будет использован последний файл из `/opt/hw-fh_backup`.
- `-l`, `--list`: показывает список файлов в директории `/opt/hw-fh_backup` готовых для восстановления из бекапа.
- `-q`, `--quiet`: восстановление из резервной копии без подтверждающего вопроса.

Восстановление данных из резервной копии на чистой машине

Для восстановления данных из резервной копии на чистой машине выполните следующие шаги:

1. Установите **Apiary** на чистую машину.
2. Скопируйте файл резервной копии в любую удобную директорию. Например `/opt/backups`.
3. Выполните команду (требуются права root):

   ```
   shell /opt/hw-fh/bin/backup-restore --file /opt/backups/
   backup_2025.08.19T20.16.02Z.tgz
   ```

4. Введите **Y**, чтобы подтвердить восстановление данных из бекапа.
5. Запустите Платформу:

   ```
   shell systemctl start hw-fh
   ```

Контроль состояния

В **Apiary** и реализована возможность получения информации, о статусе готовности внутренних компонентов (`healthcheck`).

Выполнение запроса `healthcheck` должно производиться в контексте авторизованной пользовательской сессии, в противном случае будет возвращена ошибка авторизации.

Конечная точка (`endpoint`) расположена по следующему URI: `https://apiary.server.address/api/service/health`

Пример ответа:

```
{
  "conveyServiceReady": true,
  "dataStorageReady": true,
  "redisStorageReady": true,
  "reportgenServiceReady": true,
  "applicationConnectReady": true,
  "rabbitmqServiceReady": true,
  "depotServiceReady": true
}
```

Назначение полей:

- `conveyServiceReady`: статус сервиса межсистемной коммуникации (клиент)
- `dataStorageReady`: статус реляционной СУБД
- `redisStorageReady`: статус NoSQL СУБД
- `reportgenServiceReady`: статус сервиса генерации отчетов
- `applicationConnectReady`: статус сервиса подключений типа система-система
- `rabbitmqServiceReady`: статус сервиса брокера очередей
- `depotServiceReady`: статус сервиса межсистемной коммуникации (сервер)

Диагностика проблем

Отчёт об ошибках

Большинство проблем в **Apiary** можно решить с помощью методов, указанных ниже в этой главе. Но если вам нужна дополнительная помощь по данным вопросам, или вы столкнулись с проблемой, не описанной в этой главе, обратитесь к нашей команде технической поддержки для получения дополнительной информации и регистрации проблемы.

Для того чтобы решить проблему как можно быстрее, следуйте рекомендациям:

1. Опишите условия, возникновения ошибки (что и как вы делаете);
2. Укажите что вы ожидали в результате ваших действий;
3. Укажите что вы получаете в результате ваших действий;
4. Сохраните снимки экрана с ошибкой, если это возможно;
5. Скачайте логи сервера (см. ниже);
6. Укажите версию **Apiary**. Вы можете скопировать эту информацию в окне **О нас**.
7. Отправьте всю вышеперечисленную информацию команде технической поддержки на почтовый ящик support@hexway.ru.

Журнал

В Apiary основные логи сохраняется в системный `journal` и могут быть просмотрены при помощи `journalctl`. Вы можете найти логи по этому пути: `/run/log/journal/<machine-id>/*.journal[~]`.

Вы можете настроить время глубину хранения логов при помощи настроек `journald`. Подробности по настройке можете найти, например, в справочнике вашей операционной системы:

```
shell man journald.conf
```

Чтобы увидеть логи **Apiary**, откройте консоль машины, на которой он установлен
и выполните команду:

```
journalctl -u hw-fh
```

Вы можете использовать опцию `--since` и/или `--until` для того что бы указать глубину логов для просмотра. Вы можете найти информацию по использованию этой команды и её опций, например, в справочнике вашей операционной системы:

```
shell man journalctl
```

При необходимости, сохраните логи в файл. Например, так:

```
journalctl -u hw-fh --since=-1d > system_logs_$(date +
%Y%m%d_%H%M%S).tar.gz
```

В результате будет сохранён файл с именем `system_logs_<дата>_<время>.tar.gz` который вы можете отправить на адрес support@hexway.ru.

Так же вы можете наблюдать за логами в реальном времени при помощи опции `-f`/`--follow`. Например, такая команда покажет 5 последних строк лога и будет отображать их в реальном времени:

```
journalctl -u hw-fh --follow --lines=5
```

Настройка отладочного логирования

Вы можете настроить сохранение логов для получения подробной информации о состоянии **Apiary** или ее сервисов. Существует 5 уровней логирования: `DEBUG`, `INFO`, `WARNING`, `ERROR`, и `CRITICAL`. По умолчанию, после первой установки для всех сервисов устанавливается уровень `INFO`.

Вы можете настроить различные уровни логирования для каждого сервиса по отдельности, или задать глобальные настройки для всех сервисов сразу:

- `logging.level` - глобальная настройка для всех сервисов;
- `f.engine.logging.level` - настройка для сервиса `f.engine`;
- `f.background.logging.level` - настройка для сервиса `f.background` (например, работа парсеров);
- `f.convey.logging.level` - настройка для сервиса `f.convey`.

Чтобы установить уровень логирования, нужно изменить конфигурацию.

Вы можете задать уровень логирования для каждого сервиса **Apiary**. Укажите одно из значений (`DEBUG`, `INFO`, `WARNING`, `ERROR`, или `CRITICAL`):

```
[main] f.engine.logging.level = DEBUG f.background.logging.level
= ERROR f.convey.logging.level = WARNING
```

Вы можете задать уровень логирования для всех сервисов **Apiary** и для отдельных:

```
[main] logging.level = INFO f.engine.logging.level = DEBUG
```

Не забудьте сделать `reconfig` как указано в параграфе Реконфигурация что бы изменения вступили в силу.

Статус systemd сервиса

Для диагностики состояния Apiary в первую очередь, убедитесь, что он запущен:

```
shell systemctl status hw-fh
```

Пример нормального вывода:

```
● hw-fh.service - hexway fh prod
     Loaded: loaded (/etc/systemd/system/hw-fh.service; enabled; vendor pr
     Active: active (running) since Fri 2025-03-28 17:49:36 UTC; 2 weeks 4
...
```

Пример вывода для остановленного **Apiary**:

```
○ hw-fh.service - hexway fh prod
     Loaded: loaded (/etc/systemd/system/hw-fh.service; enabled; vendor pr
     Active: inactive (dead) since Wed 2025-04-16 12:17:17 UTC; 28s ago
...
```

Важно обратить внимание, как на состояние активности сервиса (`active`/`inactive`), так и на состояния включён/выключен (`enabled`/`disabled`).

Если вы получили сообщение, например, такое:

```
Unit hw-fh.service could not be found.
```

Или обнаружили, что, по какой-то причине `systemd` сервис в настоящее время отключен, вы можете его вновь включить и активировать:

```
systemctl link /opt/hw-fh/systemd/hw-fh.service
systemctl enable hw-fh.service
systemctl daemon-reload
systemctl start hw-fh.service
```

Просмотр состояния контейнеров docker

Что бы увидеть список `docker` контейнеров выполните команду:

```
docker ps
```

Пример нормального вывода:

⚠ Разделено на четыре части по вертикали для удобства отображения.

Левая часть:

```
    CONTAINER ID   IMAGE
1.  8ebe0b7dc988   registry.hexway.io/hexway/apiary-deck:0.65.6
2.  efbd973bcca0   registry.hexway.io/hexway/apiary-engine:0.65.2
3.  2a8de6cc53d5   registry.hexway.io/hexway/convey:0.65.2
4.  6a0a90353a4f   registry.hexway.io/hexway/report-engine:0.65.1
5.  574ed2ea5a26   registry.hexway.io/hexway/depot-deck:0.65.0
6.  346739b7e3e0   registry.hexway.io/hexway/depot-service:0.65.2
7.  0b2acddc4f0a   registry.hexway.io/hexway/apiary-engine:0.65.2
8.  f52e00a63909   rabbitmq:3.11.23-management
9.  af9061df9864   postgres:16.3-alpine
10. 66d88ba73d65   postgres:16.3-alpine
11. 0ab6e96b9db8   registry.hexway.io/hexway/rmq-proxy:0.65.0
12. b77e1a9b9049   redis:7.2.5-alpine
```

Вторая часть:

```
    COMMAND                  CREATED         STATUS
1.  "/docker-entrypoint.…"   9 minutes ago   Up 9 minutes
2.  "/init-service /bin/…"   9 minutes ago   Up 9 minutes
3.  "/init-service /bin/…"   9 minutes ago   Up 9 minutes
4.  "node src/server/run…"   9 minutes ago   Up 9 minutes
5.  "/docker-entrypoint.…"   9 minutes ago   Up 9 minutes
6.  "/init-service ./run…"   9 minutes ago   Up 9 minutes
7.  "/init-service /bin/…"   9 minutes ago   Up 9 minutes
8.  "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes
9.  "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes
```

```
10. "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes
11. "/proxy-entrypoint"      9 minutes ago   Up 9 minutes
12. "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes
```

Третья часть:

```
    PORTS
1.  80/tcp, 443/tcp, 0.0.0.0:80->8080/tcp, [::]:80->8080/tcp
2.  5001/tcp
3.  5030/tcp
4.  5090/tcp
5.  80/tcp, 5040-5041/tcp
6.  5040/tcp
7.  5001/tcp
8.  4369/tcp, 5671-5672/tcp, 15671-15672/tcp, 15691-15692/tcp, 25672/tcp
9.  5432/tcp
10. 5432/tcp
11. 80/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, [::]:5672->5672/tcp
12. 6379/tcp
```

Правая часть:

```
    NAMES
1.  hw-fh-f_deck-1
2.  hw-fh-f_background-1
3.  hw-fh-f_convey-1
4.  hw-fh-r_engine-1
5.  hw-fh-d_proxy-1
6.  hw-fh-d_engine-1
7.  hw-fh-f_engine-1
8.  hw-fh-d_queue-1
9.  hw-fh-d_postgres-1
10. hw-fh-f_postgres-1
11. hw-fh-d_queueproxy-1
12. hw-fh-f_redis-1
```

В колонках `CREATED` и `STATUS` вы можете увидеть, время создания и время работы контейнеров. Обычно, время создания и время работы для каждого контейнера должно быть, примерно, одинаковым. Если вы видите, что время в `CREATED` и `STATUS` различается или в `STATUS` статус отличный от `up` это признак того, что с этим контейнером что-то не так.

Вы можете получить логи отдельного контейнера при помощи, например такой команды:

```
docker logs <container_name>
```

В качестве `container_name` укажите имя контейнера из колонки `NAMES` или идентификатор контейнера из колонки `CONTAINER ID`. Так же вы можете использовать здесь опции `--last` / `-n`, `--follow` / `-f`.

Для получения дополнительной информации по опциям выполните:

```
docker logs --help
```

В некоторых случаях достаточно изучить логи только одного контейнера для диагностики проблем.

Сетевые проходы

Необходимые сетевые проходы

Список необходимых сетевых проходов:

Целевой адрес	Целевой порт	Протокол	Назначение
registry.hexway.io	443	https	Получение образов контейнеров приложения

Сетевой проход к registry.hexway.io может не потребоваться если для инсталляции и обновления приложения используется `offline` бандл.

Опциональные сетевые проходы

Список опциональных сетевых проходов:

Источник	Целевой адрес	Порт(ы) по умолчанию	Протокол	В каких случаях необходим	Примечан
Apiary	Внешняя БД Postgresql	5432	postgresql	Хранение данных	При использова внешней БД Postgresql
Apiary	Внешняя БД Redis	6379	redis	Хранение временных данных	При использова внешней БД Redis
Apiary	Vault	8200	https	Хранение секретов доступа к внешним системам	При использова интеграции Vault
Apiary	AD сервер организации	389, 636, 3269 или 3268	LDAP(s)	Аутентификация через AD	При использова аутентифик через AD се
Apiary	SMTP сервер организации	25, 587 или 465	smpt(s)	Отправка почтовых уведомлений	При использова почтовых уведомлени

Источник	Целевой адрес	Порт(ы) по умолчанию	Протокол	В каких случаях необходим	Примечани
Apiary	Jira сервер организации	443	https	Интеграция с Jira	При использовани интеграции Jira
Jira сервер организации	Apiary	443	https	Обратная Интеграция с Jira	При использовани обратной интеграции Jira

Настройки

В этой главе описываются все основные настройки, важные для работы **Apiary**.

Настройка портов

Конфигурация портов для внешних соединений

Чтобы разрешить внешним приложениям, таким как **Hive** подключаться к **Apiary**, необходимо открыть один из следующих портов:

- `5672` - для небезопасных соединений. Чтобы обеспечить безопасное соединение, вы можете использовать такие инструменты, как *Elastic Load Balancing* (AWS) или *Google Cloud Load Balancing* (GCE) или другие инструменты, которые предоставляет ваш облачный провайдер или организовать так, как это принято в вашей организации.
- `5671` - для безопасных соединений. В этом случае необходимо настроить TLS. Для TLS потребуется доменное имя и валидный сертификат и приватный ключ. Например, вы можете использовать приложения *Let encrypt* или любого другого провайдера, который может выдать вам валидный сертификат.

Для получения доступа к пользовательскому интерфейсу и API Apiary необходимо открыть порты `80` и/или `443` (если вы не изменили их) в настройках вашего сетевого оборудования или в настройках *Load Balancer* вашего облачного провайдера, например *Elastic Load Balancing* (AWS) или *Google Cloud Load Balancing* (GCE).

Как изменить порты со стандартных на нужные вам, описано в следующем параграфе.

Кроме того, вам может потребоваться открыть все используемые порты:

- В сетевом оборудовании, если вы используете аппаратные балансировщики и/или брандмауэры в вашей организации.
- В *Security Group* (AWS) или в *Compute Engine firewall rules* (GCE) или других сервисах вашего облачного провайдера, если вы используете их для хостинга **Apiary**.
- В брандмауэре вашей операционной системы, если входящие соединения по умолчанию запрещены в вашей операционной системе.

Смена стандартных портов

Чтобы сменить стандартные порты в Apiary вам нужно изменить конфигурацию в файле `/opt/hw-fh/config/user.ini`.

Добавьте в файл следующие свойства, и укажите IP-адрес и порт.

Например, для HTTP-портов:

```
[main] f.deck.ip.expose = 127.0.0.0 f.deck.port.expose = 10001
```

Для HTTPS-портов:

```
[main] f.deck.https.ip.expose = 0.0.0.0 f.deck.https.port.expose
= 443
```

Сохраните файл и сделайте `reconfig` как указано в параграфе Реконфигурация.

Настройка доступа по https

⚠ Сразу после установки **Apiary** WebUI и API доступен только по небезопасному протоколу HTTP.

Для настройки безопасного соединения поддерживается несколько опций на выбор:

- вы можете загрузить собственные TLS сертификаты на машину с **Apiary** и выполнить настройку доступа HTTP/HTTPs;
- настроить свой HTTP или TCP Reverse proxy с TLS-терминированием перед **Apiary**.

⚠ Если вы хотите использовать самоподписанные TLS сертификаты для интеграции **Apiary** со сторонними системами, или сертификаты, выдаваемые центром сертификации внутри организации, то изучите главу Корневые TLS сертификаты.

Загрузка TLS сертификатов

Загрузите ваши собственные TLS сертификаты на машину, на которой установлен **Apiary**:

- Полную цепочку TLS сертификатов в формате PEM. Файл должен иметь имя `cert.pem` и быть доступен для чтения всем пользователям.
- Приватный ключ к этому сертификату в формате PEM. Файл должен иметь имя `key.pem` и быть доступен для чтения всем пользователям.

Вы можете использовать для этого

директории по умолчанию:

- `/opt/hw-Apiary/fssl`;
- `/opt/hw-Apiary/dpssl`;
- `/opt/hw-Apiary/qssl`.

Обратите внимание, что загруженные файлы не будут перезаписываться при обновлении на следующую версию, поэтому их можно хранить в этой директории. Обратите внимание на README.md файлы в этих директориях.

Вы также можете хранить сертификаты в другой директории, если директория по умолчанию вам не подходит. Например, в директории `/opt/my/certs/`.

Для того, что бы изменить директорию для TLS сертификата и ключа, нужно изменить конфигурацию.

Например:

```
[main] f.ssl.dir = /opt/my/certs/ q.ssl.dir = /opt/my/certs/
d.proxy.ssl.di = /opt/my/certs/
```

Не забудьте сделать `reconfig` как указано в параграфе Реконфигурация что бы изменения вступили в силу.

Настройка HTTP/HTTPs

Чтобы настроить перенаправление с HTTP на HTTPs без использования стороннего обратного прокси сервера нужно изменить конфигурацию.

Пример:

```
[main]
f.ssl.enabled = ssl_redirect
f.deck.ip.expose = 0.0.0.0
f.deck.port.expose = 80
f.deck.https.ip.expose = 0.0.0.0
f.deck.https.port.expose = 443
```

В данном примере:

- `f.ssl.enabled`: режим работы TLS. Возможные значения описаны ниже.
- `f.deck.ip.expose`: IP адрес для небезопасного HTTP-соединения (`0.0.0.0` - внешний IP адрес, `127.0.0.1` - локальный IP адрес); Обязательно для режимов `no_ssl`, `ssl_both` и `ssl_redirect`.
- `f.deck.port.expose`: номер порта для небезопасного HTTP-соединения (значение по умолчанию - 80). Обязательно для режимов `no_ssl`, `ssl_both` и `ssl_redirect`.
- `f.deck.https.ip.expose`: IP адрес для безопасного HTTPs-соединения (`0.0.0.0` - внешний IP адрес, `127.0.0.1` - локальный IP адрес). Обязательно для режимов `ssl_only`, `ssl_both` и `ssl_redirect`.
- `f.deck.https.port.expose`: номер порта для безопасного HTTPs-соединения. Обязательно для режимов `ssl_only`, `ssl_both` и `ssl_redirect`.

Режимы работы TLS `f.ssl.enabled`:

- `no_ssl` - используется только небезопасное HTTP-соединение (установлено по умолчанию);
- `ssl_both` - используется как небезопасное HTTP-соединение, так и безопасное HTTPs-соединение параллельно;

- `ssl_redirect` - используется перенаправление с небезопасного HTTP-соединения на безопасное HTTPs-соединение;
- `ssl_only` - используется только безопасное HTTPs-соединение.

Не забудьте сделать `reconfig` как указано в параграфе Реконфигурация что бы изменения вступили в силу.

Настройка обратного прокси

При необходимости **Apiary** может бежать за вашим собственным обратным прокси сервером. В этом параграфе мы рассмотрим настройку `nginx`, который будет установлен на ту же машину и будет проксировать все запросы на **Apiary**.

В этом параграфе мы сделаем:

- настроим **Apiary** что бы он не мешал обратному прокси;
- настроим `nginx` в качестве обратного прокси.

Чтобы использовать свой прокси сервер на этой же машине, необходимо изменить конфигурацию:

```
[main]
f.ssl.enabled = no_ssl
f.deck.ip.expose = 127.0.0.0
f.deck.port.expose = 10001
```

Не забудьте сделать `reconfig` как указано в параграфе Реконфигурация что бы изменения вступили в силу.

Пример №1. Конфигурации `nginx` с http и https:

```
server {
  server_name your-apiary.example.com;
  access_log /var/log/nginx/your-apiary.example.com-access.log;
  error_log /var/log/nginx/your-apiary.example.com-error.log;

  client_max_body_size 0;

  location / {
      proxy_pass http://localhost:10001;
      proxy_set_header Host $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Real-IP $remote_addr;
  }

  listen 443 ssl;
  ssl_certificate /path/to-your/certs/fullchain.pem;
  ssl_certificate_key /path/to-your/certs/privkey.pem;
}

server {
```

```
  if ($host = your-apiary.example.com) {
      return 301 https://$host$request_uri;
  }

  listen 80;
  server_name your-apiary.example.com;
  return 404;
}
```

В этом примере:

- `10001` - номер порта, который вы указали в свойстве `f.deck.port.expose` в файле `user.ini`;
- `apiary.example.com` - имя хоста или IP адрес вашей машины с установленным **Apiary**.

Пример №2. Конфигурация, `nginx`, готовая для использования с `Let's Encrypt` при помощи `certbot`.

Данную конфигурацию необходимо установить до выпуска сертификатов при помощи `certbot`.

```
server {
  listen 80;
  server_name your-apiary.example.com;
  access_log /var/log/nginx/your-apiary.example.com-access.log;
  error_log /var/log/nginx/your-apiary.example.com-error.log;

  client_max_body_size 0;

  location / {
      proxy_pass http://localhost:10001;
      proxy_set_header Host $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Real-IP $remote_addr;
  }
}
```

В этом примере:

- `10001` - номер порта, который вы указали в свойстве `f.deck.port.expose` в файле `user.ini`;
- `apiary.example.com` - имя хоста или IP адрес вашей машины с установленным **Apiary**.

TLS сертификаты для Rabbitmq

После установки, RabbitMQ по умолчанию запускается на небезопасном порту 5672. Вы можете использовать этот порт, например если **Apiary** и **Hive** установлены в безопасной локальной сети или в процессе тестирования.

Однако для производственной среды мы рекомендуем использовать безопасное соединение и настроить порт 5671 вместо небезопасного 5672. Это можно обеспечить при помощи того же самого TLS сертификата, что и для HTTPs для WebUI.

Загрузите собственные TLS полную цепочку сертификатов `cert.pem` и приватный ключ `key.pem` в формате PEM в директорию `/opt/hw-fh/qssl` (или измените этот путь в свойстве конфигурации) и измените конфигурацию для RabbitMQ:

```
[main]
q.ssl.enabled = ssl_only
rmq.ip.ssl.expose = 0.0.0.0
rmq.port.ssl.expose = 5671
rmq.client.host = my.apiary.host.com
rmq.client.port = 5671
```

Применимые свойства:

- `q.ssl.enabled` - режим работы TLS. Возможные значения указаны ниже.
- `rmq.port.expose` - номер порта для небезопасного TCP-соединения;
- `rmq.ip.expose` - IP адрес для небезопасного TCP-соединения;
- `rmq.port.ssl.expose` - номер порта для безопасного TCP-соединения;
- `rmq.ip.ssl.expose` - IP адрес для безопасного TCP-соединения;
- `rmq.client.host option` - IP адрес или имя хоста Apiary;
- `rmq.client.port` - номер порта Apiary (5671 или 5672);

Возможные значения режима работы TLS `q.ssl.enabled`:

- `no_ssl` - используется только небезопасное TCP-соединение (установлено по умолчанию);
- `ssl_both` - используются как небезопасное, так и безопасное TCP-соединение (используйте эту опцию только если вам известно, для чего вам нужно оба вида TCP-соединений);
- `ssl_only` - используется только безопасное TCP-соединение (рекомендуется для производственной среды).

⚠ Если для интеграции **Apiary** и **Hive** вы используете самоподписанные TLS сертификаты
или сертификаты, выдаваемые центром сертификации внутри организации, то изучите главу Корневые TLS сертификаты. Корневые сертификаты должны быть установлены на каждый из **Apiary** и **Hive**

Корневые TLS сертификаты

Собственные корневые сертификаты

Если вы используете самоподписанный сертификат для безопасного подключения или сертификат, выданный центром сертификации организации, или любой неглобальный SSL-сертификат, вам необходимо

вам необходимо добавить ваши корневые и/или промежуточные сертификаты для корректной работы интеграции.

Загрузите ваши корневые сертификаты в формате `PEM` или `DER` в какую либо директорию, на машине, на которой установлен **Apiary**, например, в директорию `/opt/root-certs`. Файлы могут иметь расширение `.pem`, `.crt` или `.cer`.

Укажите эту директорию
в конфигурации.

Добавьте опцию `custom.root.certs.path`. Например, так:

```
[main]
custom.root.certs.path = /opt/certs
```

Сделайте
reconfig что бы изменения вступили в силу.

⚠ Свойство `custom.root.certs.path` позволяет задать корневые сертификаты для всех сервисов **Apiary** одновременно. Если вам нужно задать раздельные корневые сертификаты, например, для интеграции между **Hive** и **Apiary** вы можете это сделать. Для дополнительной информацией вы можете обратиться к файлу `/opt/hw-fh/config/user-template.ini`

Конвертация CA сертификата из .PFX

Вам может потребоваться конвертировать CA сертификат из `.pfx`, если: * невозможно использовать глобально валидные сертификаты; * необходимо использовать внутренний Certificate Authority для выпуска сертификатов.

1. Сгенерируйте ключ TLS: `openssl pkcs12 -in your_file.pfx -nocerts -nodes -out key.pem`

2. Сгенерируйте сертификат TLS: `openssl pkcs12 -in your_file.pfx -clcerts -nokeys -out domain.pem` > Примечание: уберите опцию `-out` из команд, тогда ключ и прочая информация будет выведена на экран. > В этом случае скопируйте все содержимое со строки > `BEGIN PRIVATE KEY / BEGIN CERTIFICATE` to `END PRIVATE KEY / END CERTIFICATE` и сохраните в файл.

3. Сгенерируйте корневой сертификат: `openssl pkcs12 -in your_file.pfx -cacerts -nokeys -chain -out ca.pem` > Примечание: вы получите цепочку корневых сертификатов. > Обычно требуется только последний сертификат в этой цепочке. > Вы можете, например, открыть файл на редактирование и удалить все, кроме последнего.

4. Объедините файлы в один сертификат: `cat domain.pem ca.pem > cert.pem`

5. Скопируйте TLS сертификаты на машину с установленным **Apiary**, в директорию с TLS сертификатами: `cp cert.pem /opt/hw-fh/ssl/ cp key.pem /opt/hw-fh/ssl/`

6. Скопируйте корневой сертификат на машину в директорию с корневыми сертификатами: `cp ca.pem /opt/certs` > Примечание: в этом случае необходимо указать путь до директории с корневыми сертификатами > в файле `/opt/hw-fh/config/user.ini` с помощью опции `custom.root.certs.path = /opt/certs`.

7. Перезапустите **Apiary** (`systemctl restart hw-fh`) или выполните команду `/opt/hw-fh/bin/reconfig`, если вы изменили файл `/opt/hw-fh/config/user.ini`.

Управление лицензиями

После загрузки и установки **Apiary**, по умолчанию вы получаете версию **Сообщество**. Это означает, что некоторая функциональность будет ограничена.

> Примечание: только администратор может загрузить или удалить лицензию.

Чтобы добавить лицензию в **Apiary**:

1. Войдите в **Apiary** через UI в качестве администратора;
2. В левом меню выберите **Администрирование** > Л**Лицензия**;
3. Нажмите на кнопку **Загрузить файл лицензии** или перетащите файл лицензии в область с этой кнопкой.

После загрузки файла лицензии на вкладке Л**Лицензия** появится информация о текущей лицензии и доступных опциях.

По истечении срока действия лицензии **Apiary** автоматически вернется к версии **Сообщество**. В этом случае все существующие пользователи, проекты и т.д. останутся, но вы не сможете добавлять новые, до тех, пока снова не обновите лицензию.

События ИБ

События ИБ

В **Apiary** предусмотрена опциональная возможность сохранять все события информационной безопасности дополнительно.

Список событий ИБ, которые мы фиксируем:

- Вход пользователя в систему
- Выход пользователя из системы

- Неуспешный вход в систему
- Создание учетной записи
- Удаление учетной записи
- Блокировка (отключение) учетной записи
- Разблокировка (включение) учетной записи
- Назначение\исключение прав пользователя на объект
- Смена пароля учетной записи
- Изменение прав группы (роли) пользователей
- Исключение пользователя из состава группы (роли)
- Включение пользователя в состав группы (роли)
- Изменение Аутентификационного токена (Jira интеграция)
- Запрос к колбэку с IP вне белого списка (Jira интеграция)

Настройка сохранения события ИБ

Пример конфигурирования машин Apiary для дополнительного сохранения security логов. Нужно изменить конфигурацию:

```
[main]
#Обязательно что бы включить логи безопасности в отдельном файле.
host.install.events.enabled = enabled

#Путь к файлу лога.
#В примере указано значение по умолчанию, но вы можете изменить на своё
host.install.events.filepath = /opt/hw-fh/logs/events/events.log

#Максимальный размер файла лога для ротации.
#В примере указано значение по умолчанию.
host.install.events.filesize = 1000000

#Максимальное количество файлов лога для ротации.
#В примере указано значение по умолчанию.
host.install.events.backups = 10
```

Выполните reconfig что бы изменения вступили в силу.

Обратите внимание, что, после этого у вас появится ещё один запущенный контейнер с именем `hw-fh-a_eproc-1` из образа `registry.hexway.io/hexway/events-processor:<version>`. Вы можете увидеть это выполнив команду `docker ps`.

Пример настройки отправки событий ИБ через syslog

В случае, если нужно настроить передачу по сети на ваш сервер сбора логов или SIEM только security событий, то на машине с установленным **Apiary** нужно добавить конфигурацию `rsyslog`, как указано в примере ниже, например в файл `/etc/rsyslog.d/80-forward-eproc.conf` для передачи событий по TCP:

```
module(load="imfile" PollingInterval="10")

#Apiary security events file
input(type="imfile"
      File="/opt/hw-fh/logs/events/events.log"
      Tag="apiary-sec"
      Severity="info"
      Facility="local7")

:syslogtag, isequal, "apiary-sec:" @@rsyslog-server.corp.example.com:514
& stop
```

В этом примере: - `rsyslog-server.corp.example.com` - адрес вашего `SIEM` или `rsyslog` сервера. - 514 порт на котором ваш `SIEM` или `rsyslog` сервер принимает сообщения. - замените два символа `@@` на один `@` если нужно передавать по UDP вместо TCP.

Пример базовых событий ИБ

В примерах ниже префикс сообщения опущен и может выглядеть, например, так:

```
[<timestamp>] [INFO] SecurityLogger.Frigate: <message>
```

Здесь: - Вместо `<timestamp>` будет метка времени, например, в таком формате: `2025-04-17T13:58:14.551432+0000`. - Вместо `<message>` будет приведённое ниже сообщение.

Корректный вход в систему:

```
User root@ro.ot successfully logged in
```

Выход из системы:

```
User root@ro.ot successfully logged out
```

Некорректный вход в систему:

```
Authentication failed for user bandita: invalid login or password
```

Изменение прав на проект:

```
User root@ro.ot got OWNER access to project "My Project 1"
```

Создание локального пользователя:

```
User user1 is created
```

Получение пользователем глобальных прав в системе:

```
User user1 got EDITOR privilege
```

Добавление пользователя в группу

```
User user1 was added to the User Group group1
```

Выдача пользователю прав на проект:

```
User user1 got EDITOR access to project "My Project 1"
```

Шифрование конфиденциальных данных

Мы рекомендуем шифровать конфиденциальные данные на машине с установленным **Apiary**: - вы можете шифровать все данные на диске или отдельные директории средствами операционной системы; - вы можете шифровать пароли и ключи доступа к сторонним системам при помощи средств самого **Apiary**; - или вы можете настроить интеграцию с Vault что бы не хранить их на этой машине.

Шифрование средствами ОС

Вы можете шифровать весь диск или один из перечисленных каталогов:

- `/opt/hw-fh/data`
- `/opt/hw-fh/config`

Для шифрования вы также можете использовать:

- Программное обеспечение ОС (EncFS, CryFS, dm-crypt, etc.);
- Коммерческие приложения;
- Решения для облачных провайдеров (AWS, Azure, etc.).

Шифрование средствами приложения

При помощи этой опции можно зашифровать все пароли пользователей для внешних систем (LDAP, SMTP, JIRA, **Hive**), которые хранятся на машине с установленным **Apiary** или во внешней БД. При этом ключ шифрования будет храниться на машине, на которой установлен **Apiary**.

Вы так же можете настроить интеграцию с Vault для более надёжного хранения этих данных.

⚠ Примечание: рекомендуем активировать шифрование паролей перед тем как настраивать интеграцию с внешними системами. Если соединения уже настроены, то обратите внимание на примечание в конце этого параграфа.

Чтобы настроить шифрование паролей, необходимо изменить конфигурацию. Добавьте свойство `sensdata.encrypt.mode` и установите ему значение `AES256` (в настоящее время это единственный поддерживаемый алгоритм шифрования):

```
[main]
sensdata.encrypt.mode = AES256
```

Выполните reconfig что бы изменения вступили в силу.

⚠ Если до настройки шифрования чувствительных данных уже была настроена интеграция с внешними системами то после рестарта необходимо обновить их настройки, чтобы зашифровать существующие пароли.

Локализация интерфейса

Для интерфейса системы возможна локализация на другой язык. Это достигается путем настройки файла описания перевода интерфейса.

1. В конфигурационном файле `/opt/hw-fh/config/user.ini` в секции `[main]` добавьте свойство `f .deck.languages.dir` - путь до директории с файлами описания перевода интерфейса. Например:

   ```
   ini [main] b.deck.languages.dir = /opt/apiary-langpack
   ```

2. Создайте соответствующую директорию:

   ```
   bash mkdir /opt/apiary-langpack
   ```

3. Поместите соответствующий `json` файл описания перевода, например, `ru.json` в созданную директорию

   ```
   bash cp ./ru.json /opt/apiary-langpack
   ```

4. Выполнить реконфигурацию системы

   ```
   bash /opt/hw-fh/bin/reconfig
   ```

   При реконфигурации или перезапуске системы, из указанной директории будут подгружены все json файлы локализации с возможностью выбора языка интерфейса при входе в систему либо в настройках пользователя. Для каждого пользователя настройка используемого языка индивидуальна.

Дополнительные опции:

В файле локализации возможно определение дополнительных параметров в секции `_meta`.

- `name` - Название локализации;
- `displayName` - Отображаемое имя в селекторах выбора языка интерфейса:

- `default` - При выставлении в `true` данная локализация будет применяться по умолчанию.

Пример:

```
"_meta": {
    "name": "ru",
    "displayName": "Русский",
    "default": false
}
```

Другие настройки

Настройка профиля безопасности seccomp для Docker

Вы можете задать в явном виде профиль безопасности `seccomp` для `Docker` Для этого, нужно изменить конфигурацию.

Загрузите на машину, на которой установлен **Apiary** профиль `seccomp`, в свойстве `seccomp.profile` и укажите путь до профиля `seccomp`:

```
[main] seccomp.profile = /path/to/seccomp_profile
```

Сделайте reconfig что бы изменения вступили в силу.

Вы также можете задать раздельный профиль для каждого сервиса, входящего в **Apiary**. Более подробную информацию об опциях `seccomp.profile` вы можете найти в файле `/opt/hw-fh/config/user-template.ini`.

Конфигурация логотипа и фавикона

При необходимости можно установить собственные логотипы и фавиконы в **Apiary**:

- в **Окно логина**
- в левую навигационную панель
- в левый верхний угол **Главной страницы**
- а также в окно **О нас**.

На машине, на которую установлен **Apiary** создайте новую директорию, например:

```
mkdir /opt/my-logo
```

В эту директорию загрузите свой логотип и фавикон. В большинстве случаев вам понадобится всего три файла:

- `favicon.ico` - значок во вкладке браузера, закладках и адресной строке;
- `logo-white.svg` - логотип в левой панели навигации (размер логотипа должен быть 200 x 55 пикселей);

- `logo.svg` - логотип в верхней панели навигации и в окне логина (размер логотипа должен быть 200 x 55 px).

Нужно изменить конфигурацию добавить свойство `f.deck.logotypes.dir` в секцию `main`:

```
f.deck.logotypes.dir = /opt/my-logo
```

Выполните reconfig что бы изменения вступили в силу.

Дополнительно, используя любой генератор иконок, вы можете создать свой собственный пакет фавиконов в разных форматах для разных браузеров (например, Chrome, Safari и т. д.) и платформ (например, iOS, Android и т. д.), затем загрузить эти файлы в свой каталог. Пример:

- `apple-touch-icon.png`
- `favicon-32x32.png`
- `favicon-194x194.png`
- `android-chrome-192x192.png`
- `favicon-16x16.png`
- `safari-pinned-tab.svg`

Если вы хотите использовать разные размеры фавикона, добавьте файл `site.webmanifest` в эту же директорию с логотипом и укажите необходимые настройки. Пример:

```
{
  "icons": [
    {
      "src": "./android-chrome-36x36.png",
      "sizes": "36x36",
      "type": "image/png"
    },
    {
      "src": "./android-chrome-48x48.png",
      "sizes": "48x48",
      "type": "image/png"
    },
    {
      "src": "./android-chrome-72x72.png",
      "sizes": "72x72",
      "type": "image/png"
    },
    {
      "src": "./android-chrome-96x96.png",
      "sizes": "96x96",
      "type": "image/png"
    },
    {
      "src": "./android-chrome-144x144.png",
      "sizes": "144x144",
```

```
      "type": "image/png"
    },
    {
      "src": "./android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "./android-chrome-256x256.png",
      "sizes": "256x256",
      "type": "image/png"
    },
    {
      "src": "./android-chrome-384x384.png",
      "sizes": "384x384",
      "type": "image/png"
    },
    {
      "src": "./android-chrome-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

OpenAPI спецификация

Если вам требуется увидеть спецификацию OpenAPI (так же известную как `swagger`), но при этом не требуется доступ к ней через UI, то вы можете скачать её.

На машине, где установлен Apiary выполните команду (требуются права администратора):

```
docker cp \
  hw-fh-f_engine_1:/usr/src/app/swagger_server/swagger/swagger.yaml \
  apiary-swagger.yml
```

Вы так жде можете изучить OpenAPI спецификацию и попробовать использовать её при помощи Swagger UI. Swagger UI по умолчанию отключён, но может быть настроена для доступа. Для этого нужно изменить конфигурацию:

```
[main]
f.engine.swagger.ui.enabled = on
```

Выполните reconfig что бы изменения вступили в силу.

После старта **Apiary** откройте браузер на странице: `http://<apiary-URL>/api/ui/`

Дополнительные возможности логирования

В зависимости от политик логирования в вашей компании вам может потребоваться дополнительно настроить `systemd` модуль. Для этого в **Apiary** предусмотрены опции `system.syslog.facility` и `system.syslog.identifier`.

Пример файла `/opt/hw-fh/config/user.ini`:

```
[main]
system.syslog.facility = local5
system.syslog.identifier = my_app_logger
```

После выполнения reconfig в `systemd` модуль **Apiary** будет добавлено:

```
[Service]
...
SyslogFacility = local5
SyslogIdentifier = my_app_logger
```

Если свойства `system.syslog.facility` и `system.syslog.identifier` не заданы, то в `systemd` модуль не будет добавлено значений для `SyslogFacility` и `SyslogIdentifier` и `systemd` будет использовать значения по умолчанию.

Прочие настройки

Если вы в документации не нашли нужные вам настройки, посмотрите файл `/opt/hw-fh/config/user-template.ini`. Если вы и там не нашли нужных вам настроек, напишите в команду поддержки на почтовый ящик support@hexway.ru.

Управление пользователями

В этой главе вы можете найти информацию о добавлении пользователей и настройках по регистрации пользователей.

Если вы ищете как настроить интеграцию с LDAP так же смотрите главу LDAP в разделе Интеграции.

Добавление пользователей в систему

Добавление пользователей

Чтобы пригласить пользователей в проект, необходимо сперва добавить пользователей в **Apiary**. Все зарегистрированные пользователи могут добавлять других новых пользователей. Однако, чтобы получить доступ, все новые пользователи должны быть одобрены администраторами. Неподтвержденные пользователи не смогут залогиниться в **Apiary**.

> Примечание: только администраторы могут добавлять пользователей в Apiary.

Чтобы добавить пользователя:

1. Войдите в качестве администратора;
2. В меню слева выберите **Администрирование > Пользователи**;
3. В правом верхнем углу страница нажмите на кнопку **+ Пользователь**;
4. В окне **Новый пользователь** заполните поля: `Имя`, `Фамилия`, `Логин`, `Эл. почта`, `Пароль` и `Подтвердите пароль`;
5. Назначьте пользователю роль: **Администратор**, **Редактор** или **Только чтение**;

   > Примечание: только администраторы могут назначать пользователям права администратора.

6. Нажмите **Сохранить**. Новый пользователь появится в списке пользователей на странице **Управление пользователями**.

⚠ Примечания: 1. Пользователи LDAP автоматически появятся в списке пользователей после первого входа в **Apiary**; 2. Когда LDAP включен, вы не можете создавать новых локальных пользователей в **Apiary**, однако вы сможете добавлять бот-пользователей.

Все неподтвержденные пользователи появляются в списке на странице **Управление пользователями**.

> Примечание: для подтверждения пользователя необходимо иметь права администратора.

Чтобы подтвердить или отклонить выбранных пользователей, нажмите **Одобрить/Отклонить** или **Одобрить всех/Отклонить всех**, чтобы подтвердить всех неподтвержденных пользователей в списке.

Для редактирования пользователя, нажмите на выбранного пользователя. В окне **Пользователь** вы можете также заблокировать или удалить пользователя.

> Примечание: LDAP пользователей нельзя удалять, их можно только заблокировать. Однако, при удалении пользователя с LDAP сервера, пользователь потеряет возможность логина.

Создание бот-пользователя

Бот-пользователя можно использовать для автоматизации работы с Apiary. Например, для автоматизации поиска, автоматизации просмотра сообщений, автоматизации ответа на сообщения, или для интеграции со сторонними приложениями.

Чтобы добавить бот-пользователя на Apiary:

1. Войдите в качестве администратора;
2. В меню слева выберите **Администрирование > Пользователи**;
3. В правом верхнем углу страница нажмите на кнопку **+ Пользователь**;

4. В окне **Новый пользователь** нажмите кнопку **Создать сервисный аккаунт**;

5. Укажите логин бот-пользователя;

6. Назначьте бот-пользователю роль: Admin, Editor or Readonly;

   > Примечание: после назначения бот-пользователю роли, ее нельзя будет изменить.

7. Нажмите **Создать**. Новый бот-пользователь появится в списке пользователей на странице **Управление пользователями**;

8. Скопируйте токен бот-пользователя, для подключения бота к вашему приложению;

Вы можете проверить подключение бот-пользователя при помощи следующей команды:

```
curl --silent 'http://apiary-hostname/api/projects/' -H 'Authentication: <
```

Если все корректно, ответ сервера будет следующим:

```
[
  {
    "completionDate": "2023-03-23",
    "connectionName": null,
    "created": "2023-03-23T08:47:14.358843Z",
    "description": "",
    "groupID": "4cd961ee-50ed-4d95-9340-4d1ace7038e9",
    "hawserID": null,
    "id": "ad09eb54-31f0-44d4-b991-6bbd83bbf0f7",
    "lastIncomingPong": null,
    "lastOutgoingPing": null,
    "name": "project_name",
    "owner": {
      "firstName": null,
      "id": "e2305102-84df-45b7-a779-f028b69229fc",
      "isAdmin": true,
      "lastName": null,
      "passwordChangeRequired": false,
      "userEmail": "root@ro.ot",
      "userLogin": "root@ro.ot"
    },
    "permission": null,
    "projectType": "",
    "scope": "",
    "startDate": "2023-03-23",
    "updated": "2023-03-23T08:47:14.358863Z"
  }
]
```


Роли пользователей

В Apiary есть два типа ролей - **Глобальные** и **Проектные**.

Глобальные роли - отвечают за доступ пользователей к **Apiary**:

- **Администратор** (`administrator`) - имеет полный доступ ко всем функциональностям **Apiary**;
- **Редактор** (`editor`) - может создавать и редактировать собственные проекты; может редактировать назначенные пользователю проекты и иши, если назначена роль Editor в проекте; не может редактировать глобальные настройки **Apiary**;
- **Только чтение** (`readonly`) - не может создавать проекты и редактировать глобальные настройки **Apiary**; может редактировать назначенные пользователю проекты и иши, если назначена роль Editor в проекте.

Проектные роли - отвечают за доступ к проектам и ишам в **Apiary**:

- **Владелец** (`Owner`) - имеет полный доступ ко всем функциональностям проекта;
- **Редактор** (`Edit`) - может редактировать назначенные пользователю проекты и иши;

- **Только чтение** (`Readonly`) - может только просматривать проекты и иши.

Управление регистрацией пользователей

Apiary предоставляет возможность самостоятельной регистрации пользователей в системе.

Поведение по умолчанию:

- Пользователи могут самостоятельно регистрироваться в системе.
- Администратор должен подтвердить регистрацию пользователя в системе. То есть, все новые пользователи должны быть одобрены администраторами. Неподтвержденные пользователи не смогут залогиниться в **Apiary**.

В этом параграфе описано как можно изменить поведение по регистрации и подтверждению пользователей.

⚠ При включении интеграции с LDAP самостоятельная регистрация пользователей невозможна.

Для того что бы включить или выключить возможность самостоятельной регистрации пользователя нужно
изменить конфигурацию.

```
f.users.registration = off
```

Возможные значения:

- `on` - включение регистрации;
- `off` - отключение регистрации.

Не забудьте сделать `reconfig` как указано в параграфе Реконфигурация что бы изменения вступили в силу.

Так же вы можете включить и выключить необходимость подтверждения пользователя.

Воспользоваться этой опцией, можно только если включена регистрация пользователей (`f.users.registration = on`). Что бы изменить поведение Apiary по подтверждению пользователей нужно изменить конфигурацию:

```
f.users.confirmation = manual
```

Возможные значения:

- `auto` - автоматическое подтверждение зарегистрированных пользователей;
- `manual` - подтверждение администратором зарегистрированных пользователей вручную.

Выполните reconfig что бы изменения вступили в силу.

Многофакторная аутентификация

Apiary поддерживает многофакторную аутентификацию на основе TOTP. Чтобы активировать многофакторную аутентификацию:

1. Войдите в **Apiary** в качестве администратора;
2. В меню слева выберите **Администрирование > Пользователь**;
3. Активируйте опцию **Требуется код безопасности: вкл**.
4. Теперь, **Apiary** будет запрашивать код безопасности при каждом входе в систему:

Если пользователь потерял доступ к своему приложению-аутентификатору, нажмите на иконку для деактивации приложения. После этого, пользователю надо будет заново привязать свое приложение-аутентификатор.

Для генерации кодов безопасности, можно использовать одно из следующих приложений на выбор. Например:

- `Microsoft authenticator`
- `Google authenticator`
- `LastPass authenticator`

Расширенные возможности администрирования пользователей

Получение списка пользователей системы и их ролей

В некоторых случаях, например, для интеграции со сторонними системами, требуется получить список пользователей. В этом параграфе показан пример получения списка пользователей. Для того что бы выполнить этот http запрос вам нужно, предварительно, любым возможным получить значение куки `FSESSIONID`. Это можно сделать, например, посмотрев в инструментах разработчика или (по кнопке `F12` или `Inspect`) или выполнив `http` запрос аутентификации.

Пример запроса:

```
curl 'http://apiary.host.address/api/users/' \
    -H 'Content-Type: application/json' \
    -H 'Cookie: FSESSIONID=<session_id>'
```

Пример ответа:

Конфигурация пользовательской сессии

В **Apiary** умолчанию задано время жизни пользовательской сессии `360` минут (6 часов). То есть, через 6 часов бездействия сессия пользователя

будет оборвана и ему придётся логиниться заново. Если требуется, вы можете изменить время жизни пользовательской сессии.

Чтобы изменить время жизни пользовательской сессии в **Apiary** нужно изменить конфигурацию.

Добавьте свойство `f.engine.session.lifetime.minutes` и установите числовое значение (в минутах). Например:

```
f.engine.session.lifetime.minutes = 1
```

Сделайте reconfig что бы изменения вступили в силу.

Настройки интеграции

В зависимости от вашего рабочего процесса Apiary можно интегрировать с внешними системами управления задачами, используя токены REST API, веб-хуки и т.д.

Защищенные протоколы взаимодействия

ИС Инициатор	ИС Потребитель	Протокол взаимодействия
Apiary	Hive	AMQPS
Apiary	Active directory (AD)	LDAPS
Apiary	Hashicorp Vault	HTTPS
Apiary	JIRA	HTTPS
JIRA	Apiary	HTTPS
Apiary	ЕПС Exchange	SMTPS

Технические учетные записи

В модуле Apiary используются следующие пользователи внешних систем:

Учетные записи	Логин	Функциональность	Расположение имени пользователя	Хранение пароля
Сервисный пользователь LDAP	пользовательский	Подключение к AD	Встроенная БД платформы	приложен Hachicorp Vault
Сервисный пользователь JIRA	пользовательский	Подключение к Jira	Встроенная БД платформы	приложен Hachicorp Vault
Сервисный пользователь ЕПС Exchange	пользовательский	Подключение к ЕПС Exchange	Встроенная БД платформы	приложен Hachicorp Vault

Настройка LDAP

Настройка LDAP аутентификации

Вы можете добавлять пользователей, используя аутентификацию через LDAP.

> Примечание: добавленные пользователи могут принадлежать как к корневому домену, так и к поддоменам. Если вы хотите добавить пользователей из поддоменов, используйте Global Catalog.

- Зайдите в **Apiary** в качестве администратора;
- В меню слева выберите **Администрирование** > **LDAP**;
- Активируйте опцию **Подключение к LDAP**;

Заполните все обязательные поля:

- **LDAP протокол** - метод подключения (`ldap` или `ldaps`);
- **LDAP порт** - номер порта сервера (стандартное значение - 389);

> Примечание: когда вы добавляете пользователей из поддоменов, используйте порты Global Catalog - 3268 или 3269.

- **Адрес** - IP-адрес или имя хоста сервера;
- **Базовый DN** – каталог, в котором ведется поиск пользователя(ей). Заполните это поле, используя синтаксис LDAP, например `DC=host,DC=test,DC=domain`;
- **Имя пользователя сервисного аккаунта** - пользователь LDAP с правом просматривать содержимое ветки Base DN. Рекомендуем использовать формат `userPrincipalName` (например, `t.adm@test.domain`), но вы можете использовать и полное имя пользователя;
- **Пароль сервисного аккаунта** - пароль пользователя LDAP;

- **Фильтр пользователей** – фильтр пользователей определяет значения параметров объектов, которые будут считаться пользователями. Значения параметров необходимо вводить в соответствии с синтаксисом LDAP:
  - `(objectClass=*)` – поиск будет производиться среди всех доступных объектов;
  - `(&(objectClass=user)(loginAttr=login))` – поиск будет производиться среди объектов с соответствующими параметрами;

> Примечание: в большинстве случаев правильное значение Фильтра пользователей - `(objectClass=user)`, но если у вас нестандартный сервер LDAP, попробуйте другие значения.

- **DN группы администраторов** – определяет значения параметров объектов, которые будут считаться группами пользователей. Значения параметров необходимо вводить в соответствии с синтаксисом LDAP, например:

```
CN=U.ApiaryAdmins,CN=Users,DC=dc,DC=corp,DC=hexway,DC=com
```

- **DN группы заблокированных пользоваталей** – определяет значения параметров объектов, которые будут считаться группами заблокированных пользователей. Значения параметров необходимо вводить в соответствии с синтаксисом LDAP, например:

`CN=U.ApiaryBlocked,CN=Users,DC=dc,DC=corp,DC=hexway,DC=com`

> Примечание: в большинстве случаев правильное значение Фильтра групп - `(objectClass=group)`, но если у вас нестандартный сервер LDAP, попробуйте другие значения.

- **Атрибут имени логина** – параметр, который будет использован для аутентификации пользователей, например `sAMAccountName` – соответствует логину в формате `t.adm`;
- **Атрибут адреса электронной почты** – имя атрибута, которое содержит e-mail пользователей, например `userPrincipalName` – соответствует логину в формате `t.adm@test.domain`;

  > Примечание: если вы хотите подключиться только к поддомену, используйте логин в формате `userPrincipalName` (например, `t.adm@test.domain`).

- **Атрибут имени пользователя (first name)** – имя атрибута, которое содержит имена пользователей (например `givenName`);
- **Атрибут фамилии пользователя (last name)** – имя атрибута, которое содержит фамилии пользователей (например `sn`).

После заполнения полей, нажмите **Проверка соединения**, чтобы проверить корректность соединения с LDAP сервером.

Если настройки корректные, вы увидите сообщение **Соединение установлено**:

Нажмите на иконку `i` для просмотра доступных пользователей:

Нажмите **Сохранить**, чтобы сохранить настройки. Пользователь появится на вкладке Users после первого входа в систему.

Известные проблемы при LDAP аутентификации

- Доступна только **простая аутентификация** (`simple authentication`), **gss-api** недоступен.
- Если вы обращаетесь к домену с поддоменами, но без роли Глобального каталога (Global Catalog) (например, номер порта - 389, а значение Базы поиска - `DC=test,DC=domain`), то может возникнуть ошибка `Unprocessed Continuation Reference(s)`. Для того чтобы ошибка не возникала, укажите в Базе поиска более точное значение, например, `CN=users,DC=test,DC=domain`.

Настройка SMTP-уведомлений

Базовые настройки SMTP

Для активации SMTP:

1. Войдите в **Apiary** в качестве администратора;
2. В меню слева выберите **Администрирование** > **SMTP**;
3. Активируйте SMTP;

4. Заполните следующие поля:

   - `Адрес` - IP-адрес или имя хоста SMTP сервера;
   - `Порт` - номер порта SMTP сервера;
   - `Имя пользовтаеля` - логин отправителя;
   - `Пароль` - пароль отправителя;
   - `Безопасность` - использование протокола безопасности (NONE, STARTTLS, SSL/TLS);
   - `От Кого` - почтовый адрес, с которого вы будете получать уведомления.

5. Чтобы применить настройки, нажмите **Сохранить**.

6. Чтобы проверить правильность всех настроек SMTP, нажмите + `Проверить конфигурацию электронной почты`. С помощью этой опции вы можете отправить тестовое письмо на свой почтовый ящик:

   - `Кому` - адрес e-mail, на который будет отправлено тестовое письмо;
   - `Тема` - тема письма;
   - `Сообщение` - текст письма.

7. Нажмите **Отправить пробный e-mail**, чтобы отправить письмо.

Абсолютные ссылки

Для формирования абсолютной ссылки в письме необходимо изменить конфигурацию.

Добавьте свойство `f.deck.base.url` и укажите в нём доменное имя. Например:

```
f.deck.base.url = http(s)://Apiary.example.com:port_number
```

Что бы изменения вступили в силу нужно сделать reconfig.

Добавление пользовательских шаблонов писем

Чтобы добавить пользовательские шаблоны писем в **Apiary** на машине, на которой он установлен создайте новую директорию, например:

```
mkdir /opt/letter-templates
```

В новую директорию загрузите свой шаблон письма. Важно: имя шаблона должно быть `email_notifications.json`;

Теперь нужно изменить конфигурацию.

Добавьте свойство `f.notification.template.dir` и ней укажите путь до шаблона письма:

```
f.notification.template.dir = /opt/letter-templates
```

Что бы изменения вступили в силу нужно сделать reconfig.

Формат шаблона письма

Шаблон письма:

```
"email_new_issue":{
  "subject": "[{{product}}] New issue in project {{project.name}}: {{issue
  "body": "There is 1 new issue \n{% if project %}\n{{deck_base_url}}/proj
  },
"email_issue_status_change":{
  "subject": "{{product}}] New chat message in {{project.name}}: {{issue.n
  "body": "There is 1 new chat message \n{% if project %}\n{{deck_base_url
  }
}
```

Вы можете использовать предустановленные переменные, чтобы добавить в письмо информацию о проекте, ишах, пользователях и прочую информацию, например:

- `product` - название **Apiary**;
- `deck_base_url` - абсолютная ссылка на проект или ишью, пример:
  - `{{deck_base_url}}/projects/{{project.id}}` - абсолютная ссылка на ID проекта;
  - `{{deck_base_url}}/projects/{{project.id}}/issues?issueId={{issue.id}}` - абсолютная ссылка на ID иши;
- `user` - информация о пользователе, например:
  - `id` - ID пользователя (пример: `{{user.id}}`);
  - `login` - логин пользователя (пример: `{{user.login}}`);
  - `first_name` - имя пользователя (пример: `{{user.first_name}}`);
  - `last_name` - фамилия пользователя (пример: `{{user.last_name}}`);
  - `email` - E-mail пользователя (пример: `{{user.email}}`);
  - `is_admin` - роль пользователя - администратор (пример: `{{user.is_admin}}`);
  - `is_bot` - бот пользователь (пример: `{{user.is_bot}}`);
  - `created` - дата создания пользователя (пример: `{{user.created}}`);

- `issue` - информация об ише, например:
  - `id` - ID иши (пример: `{{issue.id}}`);
  - `project_id` - ID проекта, в котором зарегистрирована иша (пример: `{{issue.project_id}}`);
  - `name` - название иши (пример: `{{issue.name}}`);
  - `sla` - автоматически установленное значение SLA (пример: `{{issue.sla}}`);
  - `sla_manual_set` - вручную установленное значение SLA (пример: `{{issue.sla_manual_set}}`);
  - `created` - дата создания иши (пример: `{{issue.created}}`);
  - `completed` - дата устранения иши (пример: `{{issue.completed}}`);
- `project` - информация о проекте, например:
  - `id` - ID проекта (пример: `{{project.id}}`);
  - `name` - имя проекта (пример: `{{project.name}}`);
  - `description` - описание проекта (пример: `{{project.description}}`);
  - `connection_name` - имя соединения проекта (пример: `{{project.connection_name}}`);
  - `start_date` - дата начала проекта (пример: `{{project.start_date}}`);
  - `completion_date` - дата окончания проекта (пример: `{{project.completion_date}}`);
  - `scope` - общая информация о проекте (пример: `{{project.scope}}`);
  - `project_type` - тип проекта (пример: `{{project.project_type}}`);

Уведомления

В **Apiary** существуют следующие типы уведомлений:

1. Веб-уведомления - эти уведомления отображаются в Web интерфейсе при появлении соответствующих событий:
   - **Новая ишью** - добавлена новая ишью;
   - **Новое сообщение** - новое сообщение в чате иши;
   - **Изменился статус иши** - изменен статус иши.
2. E-mail уведомления - e-mail отправляется при появлении соответствующих событий. Для использования этой опции необходимо активировать SMTP уведомления:
   - **Новая ишью на эл.почту** - добавлена новая ишью;
   - **Новое сообщение на эл.почту** - новое сообщение в чате иши;
   - **Изменился статус иши на эл.почту** - изменен статус иши.

Чтобы просмотреть и настроить **Уведомления**:

1. Нажмите на иконку в правом верхнем углу страницы;
2. Нажмите на активное уведомление на вкладке **События**, и вы будете перенаправлены в ишу;
3. Перейдите на вкладку **Настройки**;

4. Включите или отключите необходимые уведомления с помощью кнопок.

Интеграция с Jira

Чтобы настроить интеграцию Apiary с Jira, выполните следующие шаги:

> Примечание: только владелец проекта или администраторы Apiary могут настраивать интеграцию.

Как настроить интеграцию с Jira:

1. Откройте свой проект;
2. В меню слева выберите **Проекты > Настройки > Интеграции**;
3. Выберите **Jira**:
4. В окне **Настроить Jira** выберите сервер Jira:
5. В зависимости от настроек вашего Jira сервера, заполните поля (все поля обязательные)
   - соединение через веб-хуки - заполните поля **Логин**, **Пароль** и **Имя соединения с Jira**:
   - соединение через токены - заполните поля **Эл. почта**, **Токен** и **Имя соединения с Jira** (смотрите статью Manage API tokens);

     > Примечание: вы можете использовать API токен для аутентификации скрипта или любого другого процесса в облачном продукте Atlassian. Вы можете сгенерировать API токен из своей учетной записи Atlassian, затем скопировать и вставить его в скрипт.
6. Нажмите кнопку **Далее**.
7. В поле **Проект** укажите проект, для которого необходимо создать соединение с Apiary;
8. В поле **Тип задач** укажите тип задачи, которая будет создана для иши в вашем проекте Jira;
   - **Обратная синхронизация** - активируйте эту опцию для синхронизации статусов ишей в Apiary со статусами задач в Jira, и укажите необходимые статусы (необязательно). В этом случае при изменении статуса задачи в Jira, в Apiary статус иши поменяется на указанный в настройках:

     > Примечание: вы можете указать пользовательские статусы для ишей проекта в Jira. Но помните, что пользовательские

статусы задач в Apiary не синхронизируются со статусами задач в Hive.

- **Синхронизировать комментарии** - активируйте эту опцию для синхронизации комментариев к ишам в Apiary и Jira (необязательно).

9. Нажмите **Завершить**, чтобы сохранить настройки. Новое соединение с Jira будет добавлено на страницу Integrations.

   Примечание: в одном проекте Apiary можно настроить интеграцию с несколькими проектами Jira одновременно. В таком случае, можно будет отправлять иши сразу в несколько проектов Jira, и читать комментарии из разных проектов Jira в одной ише.

Интеграция с Vault

В этой главе рассматривается интеграция **Apiary** с Hashicorp Vault. В частности:

• каким образом можно настроить хранение секретов в Vault;
• алгоритм переноса существующих секретов в Vault;
• особенности реализации интеграции с Vault.

Общая конфигурация интеграции с Vault

Для конфигурации работы с Vault необходимо изменить конфигурацию. В таблице ниже приведены свойства, которые поддерживаются механизмом интеграции **Apiary** и Vault.

Имя свойства	Назначение и комментарии
`vault.address`	Адрес `vault` сервера. Обязательное поле.
`vault.role_id.value`	Значение role_id. Обязательное поле.
`vault.secret_id.wrapped.file_path`	Путь к файлу с запакованным (wrapped) секретом. Обязательное поле, если секрет не распаковывается (unwrapping) автоматически сторонним способом.
`vault.secret_id.unwrapped.file_path`	Путь к файлу с распакованным (unwrapped) секретом. Обязательное поле.
`vault.secret_id.unwrap.engine_type`	Тип движка Vault. Возможные значения - kv-v1, kv-v2, approle. Дефолтное значение - kv-v2.

Имя свойства	Назначение и комментарии
	Обязательное поле, если используется запакованный (wrapped) секрет.
`vault.secret_id.unwrap.field`	Наименование поля для хранения распакованного (unwrapped) секрета. Обязательное поле, если используются kv-v1 и kv-v2. Для `approle` дефолтное значение - `secret_id`.
`vault.secret_id.unwrap.cacert`	Путь к файлу CA сертификата для доступа к vault серверу по https. Если путь не задан, то используются корневые сертификаты ОС.
`vault.secret_id.unwrap.capath`	Путь к директории с CA сертификатами для доступа к vault серверу по https. Если путь не задан, то используются корневые сертификаты ОС.
`vault.auth.mount_point`	Полный путь к логину. Может быть как однословным, напр. `my-approle-1`, так и в виде пути, напр. `my/dep/approle`

Как правило, не нужно использовать все эти свойства. В следующем параграфе приведены примеры конкретных конфигураций - вы можете выбрать наиболее подходящий для вас.

Примеры конфигураций

Пример 1. Если секрет попадает на машину с **Apiary** запакованный (wrapped) и его нужно распаковывать (unwrap) средствами **Apiary**:

```
[main]
vault.address = https://my-vault.example.com:8200
vault.auth.mount_point = my-approle-1
vault.role_id.value = 7d4714da-1c96-97be-a1ce-c19158d2a1ef
vault.secret_id.unwrapped.file_path = /opt/vault_secret/secret_id
vault.secret_id.wrapped.file_path = opt/vault_secret/wrapped_secret_id
vault.secret_id.unwrap.engine_type = approle
```

Пример 2. Если секрет попадает на машину с **Apiary** уже распакованный (unwrapped):

```
[main]
vault.address = https://my-vault.example.com:8200
vault.auth.mount_point = my-approle-1
vault.role_id.value = 7d4714da-1c96-97be-a1ce-c19158d2a1ef
vault.secret_id.unwrapped.file_path = /opt/vault_secret/secret_id
```

Не забудьте сделать `reconfig` как указано в параграфе Реконфигурация что бы изменения вступили в силу.

Распаковка секрета

Если секрет запакованный (wrapped), и **Apiary** настроен, как показано в примере 1 выше, тот этот секрет автоматически распаковывается при старте приложения. Эта операция может быть выполнена вручную при помощи утилиты `vault` в составе **Apiary**:

```
/opt/hw-fh/bin/vault unwrap
```

⚠ Примечание: для работы этой утилиты требуется утилита jq (легкий и гибкий JSON-процессор командной строки). Пожалуйста, убедитесь, что у вас установлена `jq` на машине с **Apiary**. В большинстве дистрибутивов `jq` доступен в базе пакетов.

Убедиться, что `jq`и прочие необходимые утилиты установлены в вашей ОС, вы можете проверить, введя следующую команду:

```
/opt/hw-fh/bin/vault check-config
```

Так же распаковка секрета может быть выполнена при помощи ваших средств автоматизации с использованием `/opt/hw-fh/bin/vault` или без неё. Если вы делаете это при помощи ваших средств автоматизации обратите внимание на параграф Особенности взаимодействия с Vault.

Задание путей хранения паролей в Vault для интеграций

После того как общая интеграция с Vault настроена, можно перенести хранение паролей к сервисным аккаунтам AD, SMTP сервера и Jira сервера используемых **Apiary**.

Предполагаем, что вы сохранили в Vault пароли к этим сервисным аккаунтам и, если требуется, настроили там на стороне **Vault** авторотацию. Это может быть отдельный секрет или поле в другом секрете - на ваш выбор.

Войдите в WebUI в **Apiary** как администратор, откройте настройки AD и в поле **Пароль сервисного аккаунта** введите параметры подключения к **Vault**, например, так:

```
engine=kv-v2,mount_point=hw/apiary-1,path=ad_accounts,key=ad_service
```

В данном примере:

- `kv-v2`: это название `secret engine`, которое используется в вашем **Vault**. В настоящее время в **Apiary** поддерживаются `kv-v1` и `kv-v2`.
- `hw/apiary-1`: это путь к `vault secret engine` (так же известная, как `mount point`), которое вы использовали, когда активировали этот `secret engine` в `vault`.
- `ad_accounts`: путь к секрету внутри `secret engine`, в котором хранятся ваши секреты.
- `ad_service`: имя ключа в котором хранится пароль к сервисному аккаунту.

В настоящее время все 4 поля (`engine`, `mount_point`, `path`, `key`) являются обязательными.

После того как вы изменили пароль к сервисному аккаунту к AD, выполните **Проверка соединения**.

Если **Проверка соединения** выполнена успешно это означает, что:

- Секрет для аутентификации в vault был успешно распакован (если предоставлялся запакованным) или просто успешно прочитан **Apiary**.
- При помощи указанного `role_id` и этого секрета удалось корректно залогиниться в **Vault**.
- **Vault** `secret engine` по указанному в `mount_point` пути существует и доступен на чтение для этой `approle`.
- Версия этого **Vault** `secret engine` указана корректно.
- Путь внутри этого `secret engine` указан корректно.
- Указанный ключ с паролем по указанному пути нашёлся.
- В этом ключе хранится корректный пароль к AD серверу и сервисному аккаунту удалось залогиниться в AD и получить список пользователей.

Если **Проверка соединения** оказалась неуспешной, необходимо проверить корректность введенных данных на каждом шаге, указанном выше.

Аналогичным образом вы можете задать пути для секретов для соединения с Jira и SMTP сервером.

Перенос существующего секрета в Vault

Если у вас уже была настроенная и работающая **Apiary** вам может потребоваться перенести секреты, которые сейчас в настоящее время хранятся в ini файлах в **Vault**. В данном параграфе описана последовательность действий для переноса остальных секретов.

Общая последовательность действий для каждого секрета выглядит так:

- Получить существующее значение секрета при помощи `/opt/hw-fh/bin/get-config-value` из существующего в `ini` файлах свойства (список свойств смотрите ниже).
- Создать в **Vault** секрет с этим значением. Это может быть отдельный секрет или поле в существующем секрете - на ваш выбор.
- Сформировать полный путь этого секрета в **Vault**, включая `mount-point`, `path`, `key`.
- Сохранить полный путь секрета в новое свойство (см. список ниже) в `/opt/hw-fh/config/user.ini`. Пример пути: `VAULT;engine=kv_v1,mount_point=role_v1_t,path=hw/hive»,key=cipher_key`
- Удалить старое свойство при помощи `/opt/hw-fh/bin/rm-config-value` из ini файлов.
- Для применения настроек выполните reconfig.
- Обратите внимание, что после применения настроек в файлах `local.ini` появятся новые значения для свойств, которые вы удалили. Эти значения автоматически генерируются при каждой реконфигурации системы, однако, эти значения не применяются и не являются секретами для каких-либо сервисов.

Свойства, которые нужно задать для переноса паролей и ключа шифрования в Vault:

- из `sensdata.encrypt.key` в `sensdata.encrypt.key_path`.
- из `f.postgres.password` в `f.postgres.password_path`.
- из `d.postgres.password` в `d.postgres.password_path`.
- из `rmq.password` в `rmq.password_path`.

Подробнее по каждому свойству.

Свойство `sensdata.encrypt.key_path`:

- Путь в хранилище секретов **Vault** к ключу шифрования чувствительных данных, таких как пароли соединений проектов **Hive** - **Apiary**. Если путь не задан, используется ключ шифрования, указанный в свойстве `sensdata.encrypt.key`. При некорректном указании пути, появляется сообщение об ошибке в логе приложения.
- Указывается как строка в формате: `<ENGINE>;<PATH>`, где `ENGINE` в настоящее время только `VAULT`, а `PATH`: `engine=<engine_type>,mount_point=<mount_point>,path=<path>,key=<key>`.

- Пример: `VAULT;engine=kv_v1,mount_point=role_v1_t,path=new/default,key=cipher_key`

- Примечания:

  - Используйте `get-config-value sensdata.encrypt.key` что бы получить существующий ключ шифрования.
  - После переноса ключа шифрования в **Vault** рекомендуем удалить свойство при помощи утилиты `rm-config-value sensdata.encrypt.key`.
  - Ключи шифрования для **Hive** и **Apiary** различаются, хотя свойство имеет одинаковое имя.
  - Изменение ключа шифрования для существующих коннектов не предусмотрено.

Свойство `f.postgres.password_path`:

- Путь в хранилище секретов **Vault** к паролю встроенной БД `postgresql` сервиса. Если путь не задан, используется пароль, указанный в свойстве `f.postgres.password`. При некорректном указании пути, появляется сообщение об ошибке в логе приложения.

- Указывается как строка в формате: `<ENGINE>;<PATH>`, где `ENGINE` в настоящее время только `VAULT`, а `PATH`: `engine=<engine_type>,mount_point=<mount_point>,path=<path>,key=<key>`.

- Пример: `VAULT;engine=kv_v2,mount_point=role_v2_t,path=hw/customer-portal/main_pg,key=password`

- Примечания:

  - Используйте get-config-value f.postgres.password что бы получить существующий пароль.
  - После переноса пароля в Vault рекомендуем удалить свойство при помощи утилиты rm-config-value f.postgres.password.
  - Изменение пароля в контейнере со встроенным postgresql может быть произведено вручную, если это требуется.

Свойство `d.postgres.password_path`:

- Путь в хранилище секретов **Vault** к паролю встроенной БД `postgresql depot` сервиса. Если путь не задан, используется пароль, указанный в свойстве `d.postgres.password`. При некорректном указании пути, появляется сообщение об ошибке в логе приложения.

- Указывается как строка в формате: `<ENGINE>;<PATH>`, где `ENGINE` в настоящее время только `VAULT`, а `PATH`: `engine=<engine_type>,mount_point=<mount_point>,path=<path>,key=<key>`.

- Пример: `VAULT;engine=kv_v2,mount_point=role_v2_t,path=hw/customer-portal/depot_pg,key=password`

- Примечания:
  - Используйте `get-config-value d.postgres.password` что бы получить существующий пароль.
  - После переноса пароля в Vault рекомендуем удалить свойство при помощи утилиты `rm-config-value d.postgres.password`.
  - Изменение пароля в контейнере со встроенным postgresql может быть произведено вручную, если это требуется.

Свойство `rmq.password_path`:

- Путь в хранилище секретов **Vault** к паролю встроенного брокера сообщений rabbitmq. Если путь не задан, используется пароль, указанный в свойстве `rmq.password`. При некорректном указании пути, появляется сообщение об ошибке в логе приложения.

- Указывается как строка в формате: `<ENGINE>;<PATH>`, где `ENGINE` в настоящее время только `VAULT`, а `PATH`: `engine=<engine_type>,mount_point=<mount_point>,path=<path>,key=<key>`.

- Пример: `VAULT;engine=kv_v2,mount_point=role_v2_t,path=hw/customer-portal/rmq,key=password`

- Примечания:
  - Используйте `get-config-value rmq.password_path` что бы получить существующий пароль.
  - После переноса пароля в Vault рекомендуем удалить свойство при помощи утилиты `rm-config-value rmq.password`.
  - Изменение пароля в контейнере со встроенным `rabbitmq` может быть произведено вручную, если это требуется.

Особенности взаимодействия с Vault

Как **Apiary** логинится в **Vault**:

- `role_id`: хранится на диске в ini файле и передаётся в контейнер в виде переменной среды окружения.

- Секрет для аутентификации распаковывается только (unwrap) один раз при старте **Apiary**.

- Запакованный `secret_id`: хранится на диске в директории, определённой администратором системы и используется только для распаковки (unwrap) секретов.

- Распакованный `secret_id`: хранится на диске в директории, определённой администратором системы и передаётся в контейнер как ReadOnly том (volume).

- Поскольку при старте **Apiary** запускается несколько контейнеров, и в каждом контейнере запускается несколько процессов, которым требуется логин в **Vault** файл с `unwrapped_secret_id` может быть

прочитан несколько раз и логин в **Vault** будет совершён каждым из этих процессов. Поэтому, мы рекомендуем использовать большое или бесконечное число раз, с которым **Apiary** может логиниться в **Vault**.

- При этом часть процессов (для background обработки) могут запускаться не сразу при старте **Apiary**, а по мере потребности.
- После успешного логина каждый процесс получает токен для доступа к секрету и далее работает с ним.

Поведение при истечении времени использования токена и секрета AppRole:

- При истечении времени использования токена **Apiary** выполняет автоматический повторный логин в Vault при помощи того же самого секрета.
- При невозможности повторного логина с тем же секретом производится попытка повторного чтения распакованного секрета.

Особенности поведения при ротации секрета для аутентификации в Vault:

- Повторная автоматическая распаковка запакованного секрета, в случае, если **Apiary** не смогла залогиниться в Vault с имеющимся секретом, не предусмотрен.
- Если ваша система автоматического развёртывания меняет секрет для логина в **Vault** и доставляет запакованный секрет на машину с **Apiary**, та же самая система должна будет распаковывать его, например, вызвав скрипт `/opt/hw-fh/bin/vault unwrap`.
- В случае, если распакованный секрет будет доставляться на машину другими способами (не через наш скрипт), пожалуйста, убедитесь, что `inode` этого файла не изменяется.

Пример sh скрипта для изменения содержимого файла не изменяя его inode:

```
touch "${unwrapped_secret_file}"
truncate -s 0 "${unwrapped_secret_file}"
chmod 644 "${unwrapped_secret_file}"
echo "${new_unwrapped_value}" >> "${unwrapped_secret_file}"
```

Шаблоны отчётов

Управление шаблонами отчетов

Для генерации отчетов в **Apiary** используется Docxtemplater.

Структура отчета зависит от **Схемы иш**, установленной в проекте **Hive**.

При создании нового шаблона отчета необходимо использовать технические названия полей, которые были указаны в **Схеме иш** в проекте **Hive**.

Шаблоны отчётов по умолчанию

В **Apiary** есть несколько шаблонов по умолчанию:

- `template.docx` - шаблон для схемы ишей по умолчанию;
- `universal-template.docx` - универсальный шаблон для всех доступных схем ишей.

Перед тем как загружать собственные шаблоны, рекомендуем скачать один из шаблонов по умолчанию, изменить его так как это необходимо, и загрузить шаблон обратно в **Apiary**.

⚠ Рекомендуется изменить название стандартных шаблонов перед тем как загружать их обратно в **Apiary**.

При загрузке шаблонов с существующим названием, **Apiary** автоматически изменит имя файла, добавив дату и время, например `[2022-01-06T15: 49: 23.895Z] template.docx`.

Добавление новых шаблонов отчетов

> Примечание: только администраторы **Apiary** могут загружать новые шаблоны отчетов.

Чтобы добавить новый шаблон для отчета:

1. Войдите **Apiary** в качестве администратора;
2. В левом меню выберите **Администрирование** > **Шаблоны отчётов**;
3. Нажмите **Загрузить шаблон** и загрузите шаблон с вашего компьютера;
4. Новый шаблон появится в списке шаблонов.

Теперь вы можете использовать собственные шаблоны при создании отчетов по ишам.

> Примечание: в отчет ишей экспортируются только изображения, в форматах поддерживаемых `.docx` (например, `png` и `jpeg`). Файлы в других форматах, прикрепленные к ишам, в отчет не экспортируются.

Формат шаблона отчетов

Все шаблоны отчетов настраиваемые.

> Важно! Для создания отчета необходимо использовать технические имена полей схемы ишей.

Вы можете включить в отчет:

- любой текст, который будет просто скопирован в выходной файл;
- верхние и нижние колонтитулы, оглавление, титульный лист и т.д.;
- иши со всеми полями, либо только с предопределенными полями.

Вы можете редактировать параметры стилей и создавать собственные фильтры полей в шаблонах `.docx`. Для всех полей будет сохранено форматирование, которое вы используете в вашем шаблоне отчёта.

Кроме того, вы можете использовать Markdown поля ишей в своих шаблонах отчетов.

Фильтры отчета

С помощью фильтров вы можете создать собственный вывод данных в отчете.

Общие правила фильтрации:

- Фильтр без параметров: `{myExpression | filter}`
- Фильтр с одним параметром: `{myExpression | filter:"argument1"}`
- Фильтр с несколькими параметрами: `{myExpression | filter:variable:variable2}`

Список доступных фильтров

`lower`

Конвертирует строку в нижний регистр.

```
{issue.name|lower}
```

`upper`

Конвертирует строку в верхний регистр.

```
{issue.name | upper}
```

`length`

Определяет длину коллекции (списка или строки).

```
{issues | length}
{issues | where:'totalScore >= 2' | length}
```

`prop`

Возвращает произвольное свойство объекта.

```
{issues | prop:'length'}
```

`where`

Фильтрует список элементов.

- Общий показатель риска больше или равен двум (например, 2 - средний или 3 - высокий уровень риска):

```
{#issues | where:'totalScore >= 2'} {name} {/}
```

- Общий показатель риска равен одному (где, 1 - низкий уровень риска):

```
{#issues | where:'totalScore === 1'} {name} {/}
```

- Вероятность меньше трех (например, 1 - низкий или 2 - средний уровень вероятности):

```
{#issues | where:'probabilityScore < 3'} {name} {/}
```

- Общий показатель риска равен трем, и критичность больше или равна двум (возможные значения: `&& И || ИЛИ`):

```
{#issues | where:'totalScore === 3 && criticalityScore >= 2'} {name} {/}
```

- Фильтр по любому настраиваемому полю с числовым значением:

```
{#issues | where:'myCustomFieldEstimatedLosses > 9000'} {name} {/}
```

`find`

Поиск по элементам.

```
{(issues | find:'id === 42').name}
```

`sortBy`, `sortByAsc`

Сортировка списка по возрастанию. Вы можете добавить несколько параметров:

```
{#issues | sortBy:'totalScore'} {name} {/}
{#issues | sortBy:'totalScore':'name'} {name} {/}
```

`sortByDesc`

Сортировка списка по убыванию. Вы можете добавить несколько параметров:

```
{#issues | sortByDesc:'totalScore'} {name} {/}
{#issues | sortByDesc:'totalScore':'name'} {name} {/}
```

`sumBy`

Сумма значения поля объектов из списка.

```
{issues | sumBy:'myCustomFieldBusinessExpenses'}
```

`toFixed`

Приводит число к строковому представлению с фиксированным количеством знаков после точки.

```
{myNumber | toFixed:2}
```

`json`

Выводит значение переменной в json, полезно для отладки при написании шаблона. Рекомендуется использовать с переменной `this`: отобразит текущую область видимости и все доступные переменные.

```
{myVariable | json}
{this | json}
```

`displayValue`

Отображение значения полей с перечислениями (например score, select, radio). Вызывается с системным значением поля (например 1 для score). Принимает описание поля из справочника `presetFieldsMap`.

Пример для поля `Score`:

```
{#issues}
{totalScore | displayValue:presetFieldsMap.totalScore}
{/issues}
```

Пример для кастомных полей:

• одиночный select или радио кнопки:

```
{#issues}
{myCustomField | displayValue:presetFieldsMap.myCustomField}
{/issues}
```

• множественный select:

```
{#issues}
{#myCustomListField}
{this | displayValue:presetFieldsMap.myCustomListField}
```

```
{/myCustomListField}
{/issues}
```

Стили полей Markdown

> Важно! Не меняйте идентификаторы (ID) и названия стилей в шаблонах `template.docx` и `universal-template.docx`. В настоящее время генератор отчетов поддерживает только предустановленные названия стилей.

Вы можете поменять параметры стилей непосредственно в шаблоне `.docx`, или если вы создаете собственный шаблон, используйте названия и идентификаторы стилей перечисленные ниже.

Данные стили применяются только для `Markdown` полей (`General description`, `Risk description`, `Technical description` и `Recommendations`).

ID стиля	Имя стиля	Применение	Комментарий
H1	H1	Параграф	Заголовок уровень 1
H2	H2	Параграф	Заголовок уровень 2
H3	H3	Параграф	Заголовок уровень 3
H4	H4	Параграф	Заголовок уровень 4
H5	H5	Параграф	Заголовок уровень 5
BodyExpRep	Body (Exp_Rep)	Параграф	все стандартные параметры body
BodyExpRepChar	Body (Exp_Rep) Char	Символ	символы внутри стандартного параметра body
Code	Code	Параграф	блочные строки кода
CodeChar	Code Char	Символ	строка кода
InternetLink	Internet Link	Символ	ссылки на внешние ресурсы
Emphasis	Emphasis	Символ	курсив (Italic)
StrongChar	Strong Char	Символ	жирное выделение (Bold)
PictureCaption	Picture Caption	Параграф	название рисунков и таблиц
Table style 1	Table style 1	Таблица	стиль таблиц

Стили для значений риска

Ниже приведены стили шаблонов отчетов по умолчанию для значений риска. Вы можете изменить эти стили по своему усмотрению, а также изменить идентификаторы и имена стилей.

ID стиля	Имя стиля	Применение	Комментарий
vulnrate	vulnrate	Параграф	абзац для уровня риска иши
styleRatingLow	styleRatingLow	Символ	символ для низкого уровня риска иши (зеленый)
styleRatingMedium	styleRatingMedium	Символ	символ для среднего уровня риска иши (желтый)
styleRatingHigh	styleRatingHigh	Символ	символ для высокого уровня риска иши (красный)