GsfCli

Выпуск 1.0.5

окт. 21, 2025

Содержание

1 Обзор

1.1 Назначение

Командная утилита предназначена для автоматизации работы разработчика.

`Gsf-cli` позволяет:

- Подготовить прикладной проект к работе
- Обновить зависимости необходимые для работы проекта

1.2 Требования

Для работы утилиты требуется

- `python` начиная с версии `3.9`
- `sbt` начиная с версии `1.8.2`
- `git`
-

Необходимые библиотеки:

Прописаны в `requirements.txt`:

- `bcrypt==4.0.1`
- `certifi==2023.7.22`
- `cffi==1.15.1`
- `charset-normalizer==3.2.0`
- `cryptography==41.0.3`
- `debugpy`
- `decorator==5.1.1`
- `fabric==3.1.0`
- `idna==3.4`
- `invoke==2.2.0`
- `Jinja2==3.1.2`
- `MarkupSafe==2.1.3`
- `paramiko==3.3.1`
- `prompt-toolkit==3.0.39`
- `psutil==5.9.5`
- `pycparser==2.21`
- `PyNaCl==1.5.0`
- `PyYAML==6.0.1`
- `requests==2.31.0`
- `ruamel.yaml==0.17.32`
- `ruamel.yaml.clib==0.2.7`
- `tqdm==4.65.0`
- `urllib3==2.0.4`
- `wcwidth==0.2.6`

Доп пакет:

- `python3-venv`

1.3 Установка

1. Скачайте дистрибутив `gsf-cli`

```
curl https://repo.global-system.ru/artifactory/common/ru/bitec/gsf-cli-windows/
↪SNAPSHOT/gsf-cli-windows-SNAPSHOT.zip --output gsf-cli.zip
```

   Для ручного обновления утилиты можно в каталоге c:\programs\ сделать cmd файл следующего содержания (пути подправить по необходимости)

```
curl https://repo.global-system.ru/artifactory/common/ru/bitec/gsf-cli-windows/
↪SNAPSHOT/gsf-cli-windows-SNAPSHOT.zip --output gsf-cli.zip
"C:\Program Files\7-Zip\7z.exe" x gsf-cli.zip -aoa -ogsf-cli
pause
```

2. Распакуйте архив
   Рекомендуемый путь для установки: `C:\programs\gsf-cli`
3. Установите jdk
   Установленные jdk будут искаться по адресу `C:\Program Files\Java`
4. Установите Intellij Idea
   Установленные среды будут искаться по адресу `C:\Program Files\JetBrains`

> **Внимание**
>
> В IDEA должен быть установлен плагин Scala. Подробности корректной установки смотрите в
> руководстве прикладного разработчика GlobalERP Framework.

5. Установите sbt версии 1.8.2 или выше

> **Внимание**
>
> Sbt должен быть установлен по адресу `c:\programs\sbt\`.

6. При необходимости установите `SVN` клиент
   Для авто поиска пути доступа к `svn.exe` должен быть добавлен в системную переменную `PATH`.
   Установщик `TortoiseSVN` может делать это автоматически.
7. При необходимости установите `GIT` клиент

1.4 Добавление проекта в рабочее пространство

Внимание

Если перед началом работы открыта среда разработки в общем окружении ее необходимо закрыть.

Для добавления проекта запустите скрипт `gsf-cli\links\add_project.cmd` и следуйте инструкциям мастера.
Мастер запросит необходимые параметры, и проведет подготовку проекта к работе.

Внимание

Внимательно читайте запросы мастера.

Внимание

При возникновении ошибки загрузки модулей из GitLab fatal: Unencrypted HTTP is not supported for GitHub. Ensure the repository remote URL is using HTTPS. следует выполнить команду `git config --global credential.extgit.global-system.ru.provider generic` и повторить добавление проекта.

Результат выполнения всех шагов мастера:

- `gsf-cli\workspace\dists\{project_name}\Global3se\`
  Актуальный дистрибутив сервера приложения
- `gsf-cli\workspace\sources\{project_name}\application\`
  Полностью готовый к работе проект с исходным кодом
- `gsf-cli\workspace\links\{project_name}\`
  Ярлыки быстрого запуска
- Добавленный проект становится активным

Источник проекта

Мастер конфигурации запрашивает источник определяющий откуда будет получен исходный код проекта. Формат источников:

- SVN

  ```
  https://{path}/application
  ```

- GIT

  ```
  https://{path}.git
  ```

- LXC

  ```
  lxc://{host}
  ```

  `LXC` является контейнером в котором собирается проект в системе `CI`

1.5 Работа с активным проектом

Активный проект - это проект, который будет использоваться по умолчанию в случае если он не указан явно.
Регулярные команды по работе с активным проектом смотрите в `gsf-cli\links\`

1.6 Обновление активного проекта

Для обновления зависимостей активного проекта запустите `gsf-cli\links\active_project_refresh.cmd`

1.7 Конфигурирование сервера приложения

Сервер приложения конфигурируется автоматически, для этого используется профиль конфигурации.
Пример: http://svn.bitec.ru/svn/depot/ASSource/database/pgtest/application/project/deploy/dev-win

1.8 Изменения настроек проекта

- Удалите проект командой `gsf-cli\links\delete_project.cmd`
  При вопросе об удалении файлов ответьте `нет`, что бы не выкачивать данные повторно.
- Добавьте проект с тем же именем и новыми параметрами.

Примечание

Данный подход имеет смысл только в случае если не меняется источник проекта, не считая SSL. Это позволяет избежать повторной выгрузки и компиляции проекта.

1.9 Изменения глобальных настроек

Для изменения глобальных `cli` запустите в консоли команду `gsf-cli\config.cmd configure`

1.10 Хранения паролей

Пароли сохраняются в зашифрованном виде по мастер ключу.
Мастер ключ создается автоматически при первом добавлении проекта.
Мастер добавления проекта запрашивает необходимые для дальнейшей работы пароли. Для изменения паролей смотри раздел `credential_manager`.

1.11 Горячие клавиши

- `вверх`, `вниз`
  Используется для выбора разных вариантов.
- `вправо`
  Используется для автоматического завершения команд.

2 Настройки

2.1 Настройки окружения

Путь к мастер ключу

Мастер ключ генерируется при первом запуске проекта, и используется для шифрования настроек проекта. Мастер ключ должен находится в безопасном месте и быть не доступным для других пользователей. По умолчанию мастер ключ сохраняется в рабочем каталоге пользователя.

Путь к IntelliJ IDEA

Используется для запуска среды разработки.

Путь к svn

Используется для работы с `SVN`.

Путь к sbt

Используется для работы с `SBT`.

Примечание

`Sbt` начиная с версии 1.8 для работы в режиме BSP требует отсутствие пробелов в пути. В связи с этим на данный момент требуется устанавливать sbt по адресу: `C:\programs\sbt`

Начало диапазона динамических портов

Используется для динамического выделения портов, при добавлении проекта с нестандартными портами. Позволяет одновременно запускать несколько серверов приложений.

2.2 Настройки проекта

В данной главе описываются параметры которые может спрашивать мастер, для корректного конфигурирования проектов.

Jdk

JDK с которым будет работать проект.

Url к проекту

Исходный код проекта в SVN/GIT.

Пример: `http://svn.bitec.ru/svn/depot/ASSource/database/pgtest/application`

Или: `https://extgit.global-system.ru/appdev/internal/pgtest.git`

Url к серверу приложения

Место откуда брать обновления для сервера приложения.

Пример: `ftp://ftp.bitec.ru/pub/#Global/Global3/release/Postgres/artifacts/globalserver.zip`

Или: `https://repo.global-system.ru/artifactory/general/ru/bitec/globalserver/globalserver/1.24.0/nightly/master/postgres/globalserver.zip`

Использовать стандартные порты

Если флаг сброшен, то при конфигурировании правила запуска сервера приложения порты будут динамически выделены из диапазона.

Примечание

Номера портов распечатываются при добавлении проекта. Так же их можно посмотреть в `workspace\sources\{project_name}\application\.idea\runConfigurations\Global3se.xml`

Внимание

По умолчанию это: 8080

Флаг сборки релиза

По умолчанию сброшен. Если флаг установлен сборка проекта идет в режиме релиза. Что означает что сборка запустится один раз на версию. Повторные запуски будут игнорироваться. Повторная публикация артефактов релиза запрещена. Для смены параметра смотри: `manage.py set_is_publish_release [-h]`

3 Ярлыки

Ярлыки используются для быстрого запуска часто используемых команд. Скрипты для ярлыков находятся по адресу: `gsf-cli\links`

3.1 activate_project.cmd

Активировать проект. При запуске скрипта откроется мастер, позволяющий выбрать проект для активации.

3.2 active_project_configure_idea.cmd

Сконфигурировать idea для активного проекта.

3.3 active_project_refresh.cmd

Обновить зависимости для активного проекта.

3.4 active_project_sbt.cmd

Запустить консоль SBT для активного проекта.

3.5 active_project_start_idea.cmd

Запустить среду разработки в общем окружении для активного проекта.

Внимание

Внимание, в один момент времени может быть запущена только одна среда разработки в общем окружении.

3.6 start_sep_idea.cmd

Запустить среду разработки в отдельном окружении. Это позволяет запускать несколько сред разработки одновременно. При запуске скрипта, мастер запросит проект для запуска. Рабочее окружение для работы idea сохраняется по адресу: `gsf-cli\workspace\idea\{project_name}`

3.7 add_project.cmd

Добавить проект. При этом откроется консоль с мастером для добавления проекта.

3.8 delete_project.cmd

Удалить проект. При этом откроется консоль с мастером для выбора и удаления проекта.

4 Настройка GitLab CI для сборки проектов gsf-cli

Инструкция по настройке CI процесса для автоматической сборки `applib`.

4.1 1. Настройка `gitlab-runner`

Инструкция применима к хосту сборки с `debian like` системой.

1.1 Настройка раннера в GitLab проекте

1. Перейдите в нужный проект.
2. Откройте меню: `Settings` → `CI/CD`.
3. Разверните секцию `Runners`.
4. Нажмите **«New project runner»**.
5. Укажите:
   - **Tag** — метка, по которой будет запускаться раннер.
   - **Runner description** — описание раннера.
6. **Отключите** опцию `Run untagged jobs`.

После нажатия вы перейдёте на страницу справки по регистрации gitlab-runner.

Выберите нужную операционную систему и сохраните предложенную команду регистрации.

1.2. Установка GitLab Runner

На хосте сборки выполните следующие команды:

```
# Скачивание бинарного файла GitLab Runner
sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.
↪amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

# Выдача прав на исполнение
sudo chmod +x /usr/local/bin/gitlab-runner

# Создание системного пользователя
sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

# Установка раннера как systemd-сервиса
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner

# Запуск сервиса
sudo gitlab-runner start
```

1.3. Регистрация раннера

Выполните команду регистрации, полученную на этапе `Настройка раннера в GitLab проекте`:

Пример:

```
sudo gitlab-runner register --url http://gitlablocal --token glrt-t3_m2-zeUzFHMysT3QDwmsT
```

Данная команда выдаст диалоговое окно, в котором:

- Укажите **имя** раннера (можно оставить по умолчанию).
- Выберите тип исполнителя: `shell`.
- Убедитесь, что вы используете **тот же tag**, что был указан на этапе `Настройка раннера в GitLab проекте`.

После регистрации раннер будет запускаться автоматически при старте системы и будет готов к выполнению пайплайнов.

1.4. Проверка

Для проверки работоспособности раннера, создайте минимальный `.gitlab-ci.yml` (Укажите корректный тэг раннера):

```
test-job:
  tags:
    - <your-runner-tag>
  script:
    - echo "Runner работает!"
```

Возможные проблемы

Если при запуске пайплайна, он выдает такую ошибку:

Тогда требуется:

- в файле `/home/gitlab-runner/.bash_logout` закомментировать строки:

```
if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
```

- перезапустить сервис `gitlab-runner`

```
sudo systemctl restart  gitlab-runner.service
```

4.2 2. Установка и настройка GSF-CLI на хосте сборки

Программное обеспечение, которое потребуется для сборки

- Java 21
- Sbt 1.10.7
- Утилита gsf-cli

2.1. Установка требуемых пакетов

```
sudo apt update && sudo apt install -y sudo wget git mc zip unzip
```

2.2. Подготовка директорий

Создайте рабочие каталоги:

```
sudo mkdir -p /opt/global/tmp
sudo mkdir -p /opt/global/builds
```

2.3. Загрузка необходимых компонентов

```
# GSF CLI
sudo wget -P /opt/global/tmp https://repo.global-system.ru/artifactory/common/ru/bitec/
↪gsf-cli-linux/SNAPSHOT/gsf-cli-linux-SNAPSHOT.zip

# sbt
sudo wget -P /opt/global/tmp https://github.com/sbt/sbt/releases/download/v1.10.7/sbt-1.
↪10.7.zip

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
# JDK 21 от BellSoft
sudo wget -P /opt/global/tmp https://github.com/bell-sw/Liberica/releases/download/21.0.6
↪%2B10/bellsoft-jdk21.0.6+10-linux-amd64-full.deb
```

2.4. Установка и распаковка компонентов

```
# Установка JDK
sudo apt install /opt/global/tmp/bellsoft-jdk21.0.6+10-linux-amd64-full.deb

# Распаковка SBT
sudo unzip /opt/global/tmp/sbt-1.10.7.zip -d /opt/global

# Распаковка GSF CLI
sudo unzip /opt/global/tmp/gsf-cli-linux-SNAPSHOT.zip -d /opt/global/gsf-cli
```

2.5. Установка GSF CLI

```
sudo /opt/global/gsf-cli/bin/installpkg.sh
sudo /opt/global/gsf-cli/bin/initvenv.sh
```

2.6. Настройка сборочного проекта

Конфигурационный файл

Создайте конфигурационный файл:

```
sudo nano /opt/global/builds/config.json
```

со следующим содержимым (укажите корректный `project_branch` и `project_source`)

- `project_source` — url на конфигурационный проект
- `project_branch` — ветка конфигурационного проекта

```
{
  "sbt_home": "/opt/global/sbt",
  "svn_path": "",
  "projects": [
    {
      "project_branch": "test",
      "jdk_home": "/usr/lib/jvm/bellsoft-java21-amd64/",
      "name": "main",
      "project_source": "https://extgit.global-system.ru/pgtest.git",
      "project_source_type": "vcs",
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      "publish_type": "SNAPSHOT",
      "vcs_type": "git"
    }
  ]
}
```

Детальное описание файла можно посмотреть *тут*

Регистрация приватного ключа

```
sudo /opt/global/gsf-cli/config.sh register_private_key -c /opt/global
```

Установка необходимых учётных данных

```
sudo /opt/global/gsf-cli/credential_manager.sh set -u <url> -l <login> -p <password>
```

Указывайте учетные данные для необходимых репозиториев, указанных в файле default.yaml из конфигурационного проекта.

Пример:

Если у вас закрытая среда, то ознакомьтесь с *документацией*

2.7. Активация headless-режима

Включаем `headless` режим, для отключение диалога с пользователем в процессе сборки.

```
sudo /opt/global/gsf-cli/config.sh enable_headless
```

2.8. Загрузка конфигурации

Команда для загрузки конфигурации из `config.json`.

```
sudo /opt/global/gsf-cli/config.sh load_config -f /opt/global/builds/config.json
```

2.9. Смена владельца директории

Сделайте владельцем директории с `gsf-cli`, пользователя, который был создан в шаге `Установка GitLab Runner`, в данном примере это `gitlab-runner`

```
sudo chown gitlab-runner:gitlab-runner /opt/global -R
```

4.3 3. Конфигурация пайплайна

Создайте файл `.gitlab-ci.yml` в корне проекта.

Минимальная конфигурация `.gitlab-ci.yml` для сборки проекта:

```
stages:
  - build

ssh_execute:
  stage: build
  tags:
    - <runner_tag>
  script:
    - |
      /opt/global/gsf-cli/manage.sh --all build
```

После выполнения данного пайплайна в папке `/opt/global/gsf-cli/workspace/sources/<project_name>/application/build/publish/applib` будет собранное прикладное решение.

---

5 Настройка Jenkins агента

5.1 1. Создание рабочей директории

Создайте папку `/opt/JenkinsSlave` и назначьте её владельцем пользователя, под которым Jenkins будет подключаться к агенту:

```
sudo mkdir /opt/JenkinsSlave
sudo chown qaz:qaz /opt/JenkinsSlave -R
```

5.2 2. Установка Java 17

- Обновите индексы пакетов и установите OpenJDK 17:

```
sudo apt update
sudo apt install openjdk-17-jdk -y
```

5.3 3. Настройка узла Jenkins

- Перейдите в интерфейс Jenkins и создайте новый агент (узел).
- Введите имя агента (узла) и выберите тип «Постоянный агент»

После нажатия кнопки «Создать» вы попадете в окно настройки вашего агента (узла)

Обратите внимание на следующие параметры:

- **Удалённая корневая директория:** `/opt/JenkinsSlave`
- **Использование:** «Собирать только проекты с метками, совпадающими с этим узлом»
- **Способ запуска:** Launch agents via SSH
- **Credentials:** Укажите существующие или создайте новые учетные данные

5.4 4. Расширенные настройки

В разделе «Advanced...» укажите следующие параметры при необходимости:

- **Порт:** Укажите нестандартный порт SSH, если используется не 22
- **JavaPath:** Если установлено несколько версий Java, укажите путь до нужной версии:

```
/usr/lib/jvm/java-17-openjdk-amd64/bin/java
```

5.5 5. Запуск агента

Сохраните настройки и откройте только что созданный агент. Нажмите кнопку **Launch agent**.

После успешного запуска агент будет готов к использованию.

6 Среда сборки проекта

Для сборки проекта требуется доступ к ряду репозиториев с дополнительными зависимостями (к примеру, общие библиотеки java). Есть несколько вариантов организации работы с репозиториями.

6.1 1. Сборка с использованием публичных репозиториев

- Все зависимости скачиваются из публичных репозиториев (например, Maven Central), расположенных непосредственно в интернете.

6.2 2. Сборка с использованием внутреннего прокси-репозитория

- Используется промежуточный репозиторий-прокси.
- Все зависимости запрашиваются у прокси-репозитория.
- Если зависимость не найдена — прокси делает запрос к внешним репозиториям, скачивает нужное и кэширует у себя.

Подробнее можно посмотреть в официальной документации sbt

6.3 3. Сборка в закрытой среде (изолированной)

- Среда полностью изолирована от интернета.
- Используется локальный репозиторий.
- Все зависимости предварительно вручную загружены во внутренний репозиторий.

Настройка сборки в закрытой среде.

Шаг 1. Создать файл с конфигурацией репозитория

Создайте файл `~/.sbt/repositories`:

```
mkdir  -p  ~/.sbt
nano  ~/.sbt/repositories
```

Содержимое файла:

```
[repositories]
local
maven-proxy: https://repo.global-system.ru/artifactory/common/, allowInsecureProtocol
```

`repositories` - секция, где указываются все доступные `repositories` для `sbt`. `local` - локальный репозиторий кэша. Используется для зависимостей, установленных вручную или собранных локально.
`maven-proxy: https://repo.global-system.ru/artifactory/common/, allowInsecureProtocol`

- `maven-proxy` — это имя репозитория (можно любое).
- `https://repo.global-system.ru/artifactory/common/` — URL репозитория.
- `allowInsecureProtocol` — флаг, позволяющий использовать небезопасный протокол HTTP вместо HTTPS (или с невалидным SSL-сертификатом). Начиная с версий 1.4+ `sbt` запрещает использовать HTTP. Этот флаг нужен если:
  - используется **локальный репозиторий по HTTP**, без SSL;
  - используется самоподписанный **SSL-сертификат **, и `sbt` считает его небезопасным.

Подробнее можно посмотреть в официальной документации sbt

Примечание

```
Файл `.sbt/repositories` - это конфигурация самого sbt-ланчера (загрузчика). Он нужен␣
↪sbt, чтобы понимать, откуда скачивать зависимости (библиотеки, плагины и т.д.), до␣
↪того как проект начнёт собираться.

Для сборки самого приложения необходимые файлы берутся из репозиториев указанных в␣
↪файле - `[project_name]/project/repositories/default.yaml`:

repositories:
- name: maven-common url: "https://repo.global-system.ru/artifactory/common"
- name: global-general url: "https://repo.global-system.ru/artifactory/general"
- name: app-global url: "https://repo.global-system.ru/artifactory/build-kit" isBase:␣
↪true isLoginRequired: true
```

Можно включить изоляцию репозиториев При добавлении опции `-Dsbt.override.build.repos=true` в файле проекта `application/.sbtopts` sbt будет обращаться только к репозиториям, указанным в `.sbt/repositories`

Шаг 2. Добавить файл с учётными данными (если нужен доступ по логину/паролю)

Создайте файл:

```
mkdir  -p  ~/.ivy2
nano  ~/.ivy2/.credentials
```

Содержимое:

```
realm=Artifactory Realm
host=repo.global-system.ru
user=build-user
password=very-secret-password
```

Добавьте в систему переменную окружения

`export SBT_CREDENTIALS="$HOME/.ivy2/.credentials"` с корректным путем до файла `.credentials`

Шаг 3. Добавление сертификатов. В большинстве случаев в закрытой среде потребуется настроить Java-сертификаты

1. **Получите корневой сертификат**
2. **Выполните команду:**

```
sudo keytool -import -trustcacerts \
-keystore $JAVA_HOME/lib/security/cacerts \
-storepass changeit \
-alias company-ca \
-file /путь/к/company-ca.crt
```

- `$JAVA_HOME` — путь к установленной JDK (например, `/usr/lib/jvm/bellsoft-java21-amd64`)

- `-alias` — уникальное имя сертификата в хранилище (например, `company-ca`)
- `-storepass changeit` — пароль по умолчанию для `cacerts` (если не меняли)
- `-trustcacerts` — указывает, что вы добавляете доверенный CA-сертификат

3. **Проверка добавления:**

```
keytool -list -keystore $JAVA_HOME/lib/security/cacerts -storepass changeit | grep␣
↪company-ca
```

---

7 Конфигуратор проектов

Для запуска используйте `gsf-cli\config.cmd`. Который используется для расширенного конфигурирования утилиты, в случае если не хватает ярлыков.

7.1 Commands:

```
usage: config.py [-h] cmd ...

positional arguments:
  cmd                 Команды
    full_help         Распечатать справку
    configure         Обновить конфигурацию
    load_config       Загрузить конфигурацию
    add_project       добавить проект
    delete_project    Удалить проект
    activate_project
                      Активировать проект
    enable_headless
                      Включить автономный режим
    disable_headless
                      Выключить автономный режим

options:
  -h, --help          show this help message and exit
```

Full_help

```
usage: config.py full_help [-h]

options:
  -h, --help  show this help message and exit
```

Configure

```
usage: config.py configure [-h]

options:
  -h, --help  show this help message and exit
```

Register_private_key

```
usage: config.py register_private_key [-c]

options:
  -h, --help  show this help message and exit
  -f Путь к файлу приватного ключа
```

Load_config

```
usage: config.py load_config [-h] [-f F]

Загружает конфигурацию из файла.
Конфигурация проекта содержит json файл с атрибутами:
sbt_home - местоположение sbt, если не задан sbt ищется из переменной окружения path
svn_path - местоположение svn, если не задано svn ищется из переменной окружения path
projects - массив проектов.

Атрибуты проекта:
name - имя проекта
project_source - источник проекта
jdk_home - адрес локации jdk
server_source - источник сервера приложения, игнорируется если сборка проекта идет от␣
↪комплекта сборки

options:
  -h, --help  show this help message and exit
  -f F        файл конфигурации
```

Add_project

```
usage: config.py add_project [-h]

Добавляет проект, конфигурация задается мастером создания проекта

options:
  -h, --help  show this help message and exit
```

Delete_project

```
usage: config.py delete_project [-h]

Мастер удаления проекта из конфигурации

options:
  -h, --help  show this help message and exit
```

Activate_project

```
usage: config.py activate_project [-h]

options:
  -h, --help  show this help message and exit
```

Enable_headless

```
usage: config.py enable_headless [-h]

А автономном режиме запрещено взаимодействие с пользователем.
В случае необходимости запроса пользователя будет выброшено исключение

options:
  -h, --help  show this help message and exit
```

Disable_headless

```
usage: config.py disable_headless [-h]

А интерактивном режиме возможно взаимодействие с пользователем

options:
  -h, --help  show this help message and exit
```

8 Менеджер проектов

Для запуска используйте `gsf-cli\manage.cmd`. Используется для расширенного управления проектами в случае если не хватает ярлыков.

8.1 Commands:

```
usage: manage.py [-h] [-p P] [--all] cmd ...

positional arguments:
  cmd                   Команды
    full_help           Распечатать справку
    prepare_project     Подготовить проект к работе
    refresh_server      Обновить сервер приложения
    refresh_sbt_plugin  Обновить sbt-плагин
    refresh_source      Обновить исходный код
    refresh             Обновить зависимости
    init_project        Инициализировать проект проекта
    configure_idea      Настроить idea
    set_is_publish_release
                        Установить признак публикации релиза
    publish_build_kit   Публикация комплекта сборки
    create_build_kit_release
                        Выпускает релиз комплекта сборки
    git_branch_build_kit
                        Создаёт ветку для патча комплекта сборки
    refresh_links       Обновить ярлыки
    publish             Опубликовать
    publish_sbt_plugin  Опубликовать sbt plugin
    build               Собрать проект
    test                Запустить юнит тесты
    clean               Очистить
    update_module_dependency
                        Обновление зависимостей модулей
    save_external_dependencies
                        Сохраняет набор всех внешних зависимостей решения в
                        файл
    diff_external_dependencies
                        Сравнивает набор внешних зависимостей из файла с
                        текущими от проекта

options:
  -h, --help            show this help message and exit
  -p P                  Имя проекта
  --all                 Выполнить действие для всех проектов
```

Full_help

```
usage: manage.py full_help [-h]

options:
  -h, --help  show this help message and exit
```

Prepare_project

```
usage: manage.py prepare_project [-h]

Подготавливает проект к работе, загружает сервер приложения, исходный кода, а так же␣
↪конфигурирует idea

options:
  -h, --help  show this help message and exit
```

Refresh_server

```
usage: manage.py refresh_server [-h]

Обновляет сервер приложение

options:
  -h, --help  show this help message and exit
```

Refresh_source

```
usage: manage.py refresh_source [-h]

Обновляет исходный код проекта, при необходимости делает checkout проекта

options:
  -h, --help  show this help message and exit
```

Refresh

```
usage: manage.py refresh [-h]

Обновляет зависимости

options:
  -h, --help  show this help message and exit
```

Init_project

```
usage: manage.py init_project [-h]

Инициализация проекта, создание необходимых файлов перед запуском idea

options:
  -h, --help  show this help message and exit
```

Configure_idea

```
usage: manage.py configure_idea [-h]

Конфигурация idea.
При этом происходит:
Создание конфигурации для запуска сервера приложения;
Настройка для проектов системы контроля версий.
Смотри Intellij Idea: Settings > Version Control > Directory mappings

options:
  -h, --help  show this help message and exit
```

Set_is_publish_release

```
usage: manage.py set_is_publish_release [-h]

Вызывает мастера установки признака публикации релиза.
В случае если признак установлен публикация происходит по версии
заданной в конфигурации проекта.

options:
  -h, --help  show this help message and exit
```

Publish_build_kit

```
usage: manage.py publish_build_kit [-h] [-pt {release,snapshot}]

Публикация комплекта сборки.
Версия берётся из конфигурации проекта.

options:
  -h, --help            show this help message and exit
  -pt {release,snapshot}, --publish_type {release,snapshot}
                        Тип публикации комплекта сборки. Если не указан, то
                        значение возьмётся из конфига.
```

Create_build_kit_release

```
usage: manage.py create_build_kit_release [-h]
                                          [-rt {generation,major,minor,build,patch}]

Выпускает релиз комплекта сборки.
Обрабатывается версии для корректного отображения в тегах
- Увеличивается выбранная версия и билд
- Происходит создание тега по текущей версии комплекта сборки
- Происходит commit и push изменений и тега
- Нельзя создать релиз от патча, если выбранная версия не является патчем

options:
  -h, --help            show this help message and exit
  -rt {generation,major,minor,build,patch}, --release_type {generation,major,minor,build,
↪patch}
                        Версия релиза комплекта сборки
```

Git_branch_build_kit

```
usage: manage.py git_branch_build_kit [-h]

Создаёт ветку для патча комплекта сборки.
При этом:
- Создаётся новая ветка, если её нет
- Локальная ревизия устанавливается в ветку с патчем
- Ошибка, если в project.yaml есть незакомиченные изменения

options:
  -h, --help  show this help message and exit
```

Refresh_links

```
usage: manage.py refresh_links [-h]

Обновляет ярлыки

options:
  -h, --help  show this help message and exit
```

Publish

```
usage: manage.py publish [-h]

Опубликовать комплект сборки

options:
  -h, --help  show this help message and exit
```

Publish_sbt_plugin

```
usage: manage.py publish_sbt_plugin [-h]

Опубликовать sbt plugin из комплекта сборки

options:
  -h, --help  show this help message and exit
```

Build

```
usage: manage.py build [-h]

Выполняет обновление сервера, плагина, компиляцию и публикацию

options:
  -h, --help  show this help message and exit
```

Test

```
usage: manage.py test [-h]

Выполняет юнит тестирование

options:
  -h, --help  show this help message and exit
```

Clean

```
usage: manage.py clean [-h]

Очистить

options:
  -h, --help  show this help message and exit
```

Update_module_dependency

```
usage: manage.py update_module_dependency [-h] [--force]

Обновление зависимостей модулей.

Команда актуализирует версии модулей в `project.yaml` в соответствии с требованиями в␣
↪`module-info.xml` для текущего модуля.
Проверка начинается с первого модуля в `project.yaml`.
При изменении версии какого либо модуля от которого зависит текущий модуль, происходит␣
↪повторная проверка зависимостей измененного модуля.

При нахождении расхождений в модуле подключенному по исходному коду меняется `project.
↪yaml`.
В случае если зависимость идет от комплекта сборки, выдается предупреждение.

options:
  -h, --help  show this help message and exit
  --force     Актуализирует 'project.yaml' не спрашивая пользователя
```

Save_external_dependencies

```
usage: manage.py save_external_dependencies [-h] [-f [FILE]]

options:
  -h, --help            show this help message and exit
  -f [FILE], --file [FILE]
                        Файл, в который необходимо сохранить список
```

Diff_external_dependencies

```
usage: manage.py diff_external_dependencies [-h] [-f [FILE]]

options:
  -h, --help            show this help message and exit
  -f [FILE], --file [FILE]
                        Файл для сравнения, в котором хранится список внешних
                        зависимостей
```

9 Менеджер учетных данных

Для запуска используйте `gsf-cli\credential_manager.cmd`. Используется для управления паролями для доступа к репозиторию

9.1 Commands:

```
usage: credential_manager.py [-h] cmd ...

positional arguments:
  cmd         Команды
    full_help
                Распечатать справку
    get       Загрузить конфигурацию
    show      Отобразить учетные данные
    git       git credential-helper
    set       Задать учетные данные по протоколу

options:
  -h, --help  show this help message and exit
```

Full_help

```
usage: credential_manager.py full_help [-h]

options:
  -h, --help  show this help message and exit
```

Get

```
usage: credential_manager.py get [-h] [-u U]

Получает учетные данные для заданного url.
Учетные данные предоставляются в поток вывода в формате json

options:
  -h, --help  show this help message and exit
  -u U        url для которого надо получить учетные данные
```

Show

```
usage: credential_manager.py show [-h]

Отображает сохраненные учетные данные

options:
  -h, --help  show this help message and exit
```

Git

```
usage: credential_manager.py git [-h] command

Реализует credential-helper для git

positional arguments:
  command

options:
  -h, --help  show this help message and exit
```

Set

```
usage: credential_manager.py set [-h] [-u U] [-l L] [-p P]

Задает учетные данные для заданного url, поиск будет идти по началу строки.

options:
  -h, --help  show this help message and exit
  -u U        Url для которого надо задать учетные данные
  -l L        Имя пользователя
  -p P        Пароль
```

10 Утилита для git

Используется для автоматизации выпуска релизов и переключения между ветками в проекте решения.

Для запуска используйте `gsf-cli\gsf_git.cmd`.

Совет

Для удобной работы из консоли решения используйте алиас `gsfp_git.cmd`.

Консоль решения можно открыть ярлыком `workspace/links/[project_name]/start_shell.cmd`

Алгоритм выпуска релизов:

1. Откройте консоль решения для выпуска релиза
2. Выпустите релиз модуля
   В каталоге модуля выполните `gsfp_git create_release`
3. Откройте консоль решения для релиза

Совет

Для переключение решения в релизную ветку в каталоге решения выполните `gsfp_git switch release`

4. Переключите релиз
   Для этого в каталоге модуля выполните `gsfp_git switch [release_name]`
5. Сделайте `commit` и `push` для проекта решения

10.1 Commands:

```
usage: gsf_git.py [-h] [-p P] [-w W] [-m M] [-s] [--all_modules] cmd ...

positional arguments:
  cmd                   Команды
    full_help           Распечатать справку
    refresh             Обновить исходный код решения из git репозитория
    create_generation   Выпустить поколение
    create_major        Выпустить мажорную версию
    create_minor        Выпустить минорную версию
    create_build        Выпустить билд
    create_patch_branch
                        Создаёт ветку для патча
    create_patch        Выпустить патч
    switch              Переключится на другую ветку
    update_to_last_tag
                        Актуализирует модули до последних версий тегов
    switch_to_last_tag
                        Переключает модули на последних версий тегов
    status              Отобразить статус
    version_info        Отобразить информацию по версиям

options:
  -h, --help            show this help message and exit
  -p P                  Имя проекта
  -w W                  Рабочий каталог
  -m M                  Имя модуля
  -s                    Решение
  --all_modules         Все модули
```

Full_help

```
usage: gsf_git.py full_help [-h]

options:
  -h, --help  show this help message and exit
```

Refresh

```
usage: gsf_git.py refresh [-h]

Обновляет исходный код решения и модулей git

options:
  -h, --help  show this help message and exit
```

Create_generation

```
usage: gsf_git.py create_generation [-h] [-m M [M ...]]

Выпускает поколение.
При этом:
- Обрабатывается версии для корректного отображения в тегах
- Увеличивается текущее поколение и билд версии модулей
- Происходит создание тегов по текущим версиям модулей
- Происходит commit и push изменений и тегов
- Нельзя создать поколение от патча

options:
  -h, --help    show this help message and exit
  -m M [M ...]  Список модулей для обновления
```

Create_major

```
usage: gsf_git.py create_major [-h] [-m M [M ...]]

Выпускает мажорную версию.
При этом:
- Обрабатывается версии для корректного отображения в тегах
- Увеличиваются текущие мажорная и билд версии модулей
- Минорная и релизная версия зануляются
- Происходит создание тегов по текущим версиям модулей
- Происходит commit и push изменений и тегов
- Нельзя создать мажорную версию от патча

options:
  -h, --help    show this help message and exit
  -m M [M ...]  Список модулей для обновления
```

Create_minor

```
usage: gsf_git.py create_minor [-h] [-m M [M ...]]

Выпускает минорную версию.
При этом:
- Обрабатывается версии для корректного отображения в тегах
- Увеличиваются текущие минорная и билд версии модулей
- Релизная версия зануляется
- Происходит создание тегов по текущим версиям модулей
- Происходит commit и push изменений и тегов
- Нельзя создать минорную версию от патча

options:
  -h, --help    show this help message and exit
  -m M [M ...]  Список модулей для обновления
```

Create_build

```
usage: gsf_git.py create_build [-h] [-m M [M ...]]

Выпускает билд.
При этом:
- Обрабатывается версии для корректного отображения в теге
- Увеличивается текущие билд версии модулей
- Происходит создание тегов по текущим версиям модулей
- Происходит commit и push изменений и тегов
- Нельзя создать билд от патча

options:
  -h, --help    show this help message and exit
  -m M [M ...]  Список модулей для обновления
```

Create_patch_branch

```
usage: gsf_git.py create_patch_branch [-h] [-m M]

Создаёт ветку для патча.
При этом:
- Создаётся новая ветка, если её нет
- Локальная ревизия устанавливается в ветку с патчем
- Ошибка, если есть незакоммиченные изменения
- Ошибка, если команда вызвана сразу на несколько модулей

options:
  -h, --help  show this help message and exit
  -m M        Модуль по которому создаётся ветка с патчем
```

Create_patch

```
usage: gsf_git.py create_patch [-h] [-m M [M ...]]

Выпускает патч.
При этом:
- Обрабатывается версия для корректного отображения в теге
- Увеличивается текущая версия модуля по патчу
- Происходит создание тегов по текущим версиям модулей

options:
  -h, --help    show this help message and exit
  -m M [M ...]  Список модулей для обновления
```

Switch

```
usage: gsf_git.py switch [-h] (-branch BRANCH | -mb MB [MB ...])

Переключает репозиторий на другую ветку.
При запуске от решения:
- Изменяется ветка в настройках проекта
- Происходит обновление репозиториев

При запуски от модуля:
- Изменяется ветка в файле `project.yaml`
  Внимание: Данные изменения не попадают в commit
- Происходит обновление исходного кода по модулю

options:
  -h, --help       show this help message and exit
  -branch BRANCH
  -mb MB [MB ...]  <module_name>:<branch_name>
```

Update_to_last_tag

```
usage: gsf_git.py update_to_last_tag [-h] [-m M [M ...]]

Актуализирует модули до последних версий тегов.
Модули, в которых указана ветка, а не тег - игнорируются.
При этом:
- Изменяется ветка в файле `project.yaml`
  Внимание: Данные изменения не попадают в commit
- Происходит обновление исходного кода по модулю

options:
  -h, --help    show this help message and exit
  -m M [M ...]  Список модулей для обновления
```

Switch_to_last_tag

```
usage: gsf_git.py switch_to_last_tag [-h] [-m M [M ...]]

Переключает модули на последние версии тегов.
При этом:
- Изменяется ветка в файле `project.yaml`
  Внимание: Данные изменения не попадают в commit
- Происходит обновление исходного кода по модулю

options:
  -h, --help    show this help message and exit
  -m M [M ...]  Список модулей для обновления
```

Status

```
usage: gsf_git.py status [-h]

Отображает информацию о состоянии решения и модулей.
Позволяет увидеть список модулей по которым необходимо сделать commit или push.
Решение в списке обозначено символом `.`

options:
  -h, --help  show this help message and exit
```

Version_info

```
usage: gsf_git.py version_info [-h]

Отображает информацию по версиям решения и модулей.
Решение обозначается символом `.`

options:
  -h, --help  show this help message and exit
```

11 Реестр используемых библиотек

Реестр используемых библиотек - это набор всех внешних зависимостей решения. Формат набора библиотек: <Вендор>:<Наименование>:<Версия>

11.1 Сохранение набора используемых библиотек

Что бы сохранить набор используемых библиотек, необходимо запустить батник `.../gsf-cli/workspace/links/<project>/save_external_dependencies.cmd` и указать файл, куда необходимо сохранить данные.

Внимание

Перед вызовом батника необходимо убедиться, что выполнен `reload sbt`.

11.2 Сравнение набора внешних зависимостей

Что бы сравнить набор внешних зависимостей текущего решения с набором из файла, необходимо запустить батник `.../gsf-cli/workspace/links/<project>/diff_external_dependencies.cmd` и указать файл, в котором находятся внешние зависимости.

В результате работы команды будет выведено два списка:

- «Новый зависимости» - зависимости, которые есть в решении, но отсутствуют в файле.
- «Устаревшие зависимости» - зависимости, которых нет в решении, но присутствуют в файле.

Если зависимости не отличаются, то будет выведено `Внешние зависимости идентичны`.

Внимание

Перед вызовом батника необходимо убедиться, что выполнен `reload sbt`.

12 Логирование в проекте

12.1 Общий обзор

В проекте реализована система логирования, позволяющая фиксировать ошибки и события, возникающие в процессе выполнения команд. Логирование выполняется в автоматическом режиме и не требует дополнительной настройки со стороны пользователя.

12.2 Структура логирования

Логи в проекте сохраняются в директории `workspace/logs`, которая создается автоматически при первом запуске команды. Логирование организовано следующим образом:

- **Логи командной строки** – если во время выполнения команды в терминале возникает ошибка, она автоматически перехватывается и записывается в файл `cmd_error_log.txt`.
- **Общие логи проекта** – записываются в файлы, названные в соответствии с текущей датой (`YYYY-MM-DD.log`).
- **Хранение логов** – файлы логов сохраняются за последние 10 дней, более старые файлы автоматически удаляются.

12.3 Пример структуры каталога логов

```
project_root/
├── workspace/
│   ├── logs/
│   │   ├── cmd_error_log.txt
│   │   ├── 2025-03-01.log
│   │   ├── 2025-03-02.log
│   │   ├── ...
│   │   ├── 2025-03-10.log
```

13 Конфигурационные файлы проекта

13.1 Пример содержимого config.json

```
{
  "sbt_home":  "/opt/global/sbt",
  "svn_path":  "",
  "concurrent_module_updates": 1,
  "projects":  [
  {
  "project_branch":  "<branch>",
  "jdk_home":  "/usr/lib/jvm/bellsoft-java8-amd64/",
  "name":  "<project_name>",
  "project_source":  "<project_url>",
  "project_source_type":  "vcs",
  "publish_type":  "SNAPSHOT",
  "vcs_type":  "git"
  "server_source": "<server_url>"
  }
  ]
}
```

Детальное описание полей конфигурации:

- `sbt_home`: Путь к установленному SBT (Scala Build Tool). если не задан sbt ищется из переменной окружения path
- `concurrent_module_updates`: Количество модулей, которые можно скачивать одновременно. По умолчанию - 20. Поставьте 1 для перевода в более надежный, но и более медленный однопоточный режим.
- `svn_path`: Путь к SVN (если используется) если не задан svn ищется из переменной окружения path
- `projects`: Список проектов, где каждый проект:
  - `project_branch`: Название ветки для git (опционально, по умолчанию `main`).
  - `jdk_home`: Путь к JDK (опционально).
  - `name`: Имя проекта (обязательное поле).

- `project_source`: Источник проекта, например.
  - Если начинается с `lxc://`, выбрасывается исключение («Not implemented»).
  - Иначе считается `vcs`, и если это `git`, может указываться ветка.
- `project_source_type`: Тип источника проекта.
- `publish_type`: Тип публикации (опционально, строка).
- `vcs_type`: Указывает, **какая система управления версиями** используется для проекта
- `server_source`: Источник сервера приложения, игнорируется если сборка проекта идет от комплекта сборки