GsfCli

Выпуск 1.0.3

апр. 25, 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 Установка

1. Скачайте дистрибутив `gsf-cli`

   Внимание:

   ```
   curl https://repo.global-system.ru/artifactory/common/ru/bitec/gsf-cli/SNAPSHOT/
   ↪gsf-cli-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/SNAPSHOT/
   ↪gsf-cli-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 Global3 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 путь доступа к `svn.exe` должен быть добавлен с системную переменную `path`.
   Установщик `TortoiseSVN` может делать это автоматически
7. При необходимости установите git клиент

1.3 Добавление проекта в рабочее пространство

> **Внимание:** Если перед началом работы открыта среда разработки в общем окружении ее необходимо закрыть.

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

> **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 становиться 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

1.4 Работа с активным проектом

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

1.5 Обновление активного проекта

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

1.6 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

1.7 Изменения настроек проекта

- Удалите проект командой `gsf-cli\links\delete_project.cmd`
  При вопросе об удалении файлов ответьте `нет`
- Добавьте проект с тем же именем и новыми параметрами

Примечание: Данный подход имеет смысл только в случаи если не меняется источник проекта. Это позволяет избежать повторной выгрузки и компиляции проекта

1.8 Изменения глобальных настроек

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

1.9 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

1.10 Горячии клавиши

- `вверх`, `вниз`
  Используется для выбора разных вариантах
- `вправо`
  Используется для авто завершения команды

2 Настройки

2.1 Настройки окружения

Путь к мастер ключу

Мастер ключ генерируется при первом запуске проекта, и используется для шифрование 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 Inteliji Idea

Используется для запуска среды

Путь к svn

Используется для работы с `svn`

Путь к sbt

Используется для работы с `sbt`.

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

Url к серверу приложения

Место откуда брать обновления для сервера приложения. Пример: `ftp://ftp.bitec.ru/pub/#Global/Global3/release/Postgres/artifacts/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`

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 запуски 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 Непрерывная интеграция

`Gsf-cli` может быть встроена в конвейер непрерывной интеграции для публикации решений и комплектов сборки.

4.1 Программное обеспечение которое потребуется для сборки:

1. Виртуальная машина или контейнер на базе linux
2. Java 8
3. Sbt
4. Утилита gsf-cli

4.2 Пример шагов по настройки сборщика:

1.Установите sbt (например в /home/worker/app/sbt)

1. Скачать и распаковать утилиту gsf-cli в каталог пользователя (например /home/worker/workspace/build/gsf-cli)
2. Заполнить конфигурационный файл утилиты gsf-cli для проекта (путь ~/workspace/build/config.json) Привет файла:

```
{
"sbt_home": "/home/worker/app/sbt",
"svn_path": "",
"projects": [
{
"git_branch": "main",
"jdk_home": "/usr/lib/jvm/bellsoft-java8-full-amd64/",
"name": "global-erp",
"project_source": "https://someproject.git",
"project_source_type": "vcs",
"publish_type": "SNAPSHOT",
"vcs_type": "git"
  }
 ]
}
```

4.3 Настройка в linux

1. Перейдите в каталог `gsf-cli`

```
cd ~/workspace/build/gsf-cli
```

1. Установите необходимые пакеты

```
sudo bin/installpkg.sh
```

1. Установите необходимые зависимости

```
bin/initvenv.sh
```

1. Если требуется авторизация в репозиториях

```
./credential_manager.sh set -u https://git.global-system.ru  -l username -p password_or_
↪token

./credential_manager.sh set -u https://repo.global-system.ru -l username -p password
```

1. Переведите утилиту в автономный режим

```
./config.sh enable_headless
```

1. Загрузите конфигурацию

```
./config.sh load -f config.json
```

1. Запустите сборку

```
./manage.sh --all build
```

4.4 Сборка проекта

Алгоритм сборки:

1. Если необходимо, загрузить исходный код проекта.
2. Если необходимо, загрузить сервер приложения \

Сервер приложения будет загружен либо из конфигурации проекта либо из настройки проекта.

Настройка проекта может перекрывать конфигурацию в случаи если сервер берется не из комплекта сборки.

Если решение собирается из комплекта сборки перекрытие запрещено так как это может привести к конфликтам.

1. Если необходимо, загрузить плагин
2. Скомпилировать проект
3. Если необходимо опубликовать артефакты \

Необходимость публикации определяется настройками в `project.yaml` и конфигурацией `gsf-cli`

4.5 Сборка релиза

Релиз собирается только 1 раз, после успешной сборки повторный запуск игнорирует изменения до тех пор пока

не изменится версия релиза.

Это гарантирует неизменность артефактов в репозитории.

В случаи если допустить изменяемость релизов в репозитории невозможно гарантировать корректную

доставку артефактов так как maven загружает релиз только 1 раз и далее не проверяет его на изменения.

4.6 Сборка snapshot

Snapshot версия собирается и публикуется на каждый запуск.

Совет: Для обновление перевыпущенных артефактов воспользуйтесь командой `sbt update; updateClassifiers`

Подробнее смотрите управления зависимостями в sbt

---

5 Конфигуратор проектов

Для запуска используйте `gsf-cli\config.cmd`. Используется для расширенного конфигурирования утилиты в случаи если не хватает ярлыков.

5.1 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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                    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
```

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 случаи 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
```

6 Менеджер проектов

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

6.1 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_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 случаи 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 случаи 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
```

7 Менеджер учетных данных

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

7.1 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
```

8 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

8.1 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
```

9 Реестр используемых библиотек

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

9.1 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`.

9.2 Сравнение набора внешних зависимостей

Что бы сравненить 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`.

10 Логирование в проекте

10.1 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.

10.2 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.

10.3 Пример структуры каталога логов

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