

# Руководство пользователя Zaplet

## Содержание

1. [Введение](#)
2. [Базовое использование](#)
  - [HTTP-запросы](#)
    - [GET-запрос](#)
    - [POST-запрос](#)
    - [PUT-запрос](#)
    - [DELETE-запрос](#)
    - [PATCH-запрос](#)
    - [HEAD-запрос](#)
    - [OPTIONS-запрос](#)
  - [Форматирование вывода](#)
3. [Работа со сценариями](#)
  - [Выполнение сценария](#)
4. [Продвинутые возможности](#)
  - [Заголовки запросов](#)
  - [Тайм-ауты](#)
5. [Логирование](#)
  - [Настройка логирования](#)
6. [Устранение неполадок](#)

## Введение

Zaplet — это кроссплатформенная утилита командной строки, написанная на C++20, предназначенная для тестирования REST API и создания/воспроизведения тестовых сценариев. Zaplet позволяет выполнять HTTP-запросы различных типов, настраивать заголовки, отправлять данные и анализировать ответы в удобном формате.

Основные возможности Zaplet:

- Выполнение HTTP-запросов (GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS)
- Воспроизведение тестовых сценариев из YAML-файлов
- Поддержка переменных в сценариях
- Условное выполнение шагов
- Гибкая система валидации ответов
- Удобное форматирование вывода (JSON, YAML, табличный вид)
- Настраиваемая система логирования

## Базовое использование

Zaplet предоставляет интерфейс командной строки для выполнения HTTP-запросов и запуска сценариев. Общий формат команды:

```
zaplet <команда> [опции]
```

Для получения справки по доступным командам:

```
zaplet --help
```

## HTTP-запросы

Zaplet поддерживает все основные HTTP-методы. Ниже приведены примеры использования каждого из них.

### GET-запрос

```
zaplet get https://api.example.com/users
```

Дополнительные параметры:

- -H, --header - добавление HTTP-заголовка (можно указать несколько раз)
- -t, --timeout - таймаут запроса в секундах (по умолчанию 30)

Пример с заголовком авторизации:

```
zaplet get https://api.example.com/users -H "Authorization: Bearer token123"
```

### POST-запрос

```
zaplet post https://api.example.com/users -d '{"name": "John", "email": "john@example.com"}'
```

Дополнительные параметры:

- -H, --header - добавление HTTP-заголовка
- -d, --data - данные для отправки в теле запроса
- --content-type - тип содержимого (по умолчанию "application/json")
- -t, --timeout - таймаут запроса в секундах

### PUT-запрос

```
zaplet put https://api.example.com/users/1 -d '{"name": "John Updated", "email": "john@example.com"}'
```

Параметры аналогичны POST-запросу.

### DELETE-запрос

```
zaplet delete https://api.example.com/users/1
```

Параметры:

- -H, --header - добавление HTTP-заголовка
- -t, --timeout - таймаут запроса в секундах

## PATCH-запрос

```
zaplet patch https://api.example.com/users/1 -d '{"name": "John Patched"}'
```

Параметры аналогичны POST-запросу.

## HEAD-запрос

```
zaplet head https://api.example.com/users
```

Параметры:

- -H, --header - добавление HTTP-заголовка
- -t, --timeout - таймаут запроса в секундах

## OPTIONS-запрос

```
zaplet options https://api.example.com/users
```

Параметры:

- -H, --header - добавление HTTP-заголовка
- -t, --timeout - таймаут запроса в секундах

## Форматирование вывода

По умолчанию Zaplet форматирует вывод в YAML. Можно изменить формат вывода с помощью глобальной опции --format следующим образом:

```
zaplet --format json get https://api.example.com/users
```

Поддерживаемые форматы:

- json - JSON
- yaml - YAML (по умолчанию)
- table - табличный вид

## Работа со сценариями

Zaplet позволяет выполнять ранее созданные сценарии тестирования API. Сценарии хранятся в файлах с расширением .zpl и позволяют запускать последовательность HTTP-запросов с одной командой.

Подробная документация по созданию сценариев находится в отдельном файле `scenario_writing_guide.md`.

## Выполнение сценария

Для выполнения сценария используйте команду `play`:

```
zaplet play my_scenario.zpl
```

С переменными (переопределение переменных из файла сценария):

```
zaplet play my_scenario.zpl -v base_url=https://api.staging.example.com -v  
auth_token=test_token
```

Переменные, передаваемые через командную строку, имеют приоритет над переменными, определенными в файле сценария.

## Продвинутые возможности

### Заголовки запросов

Для добавления HTTP-заголовков к запросу используйте опцию `-H` или `--header`:

```
zaplet get https://api.example.com/users -H "Authorization: Bearer token" -H  
"Accept: application/json"
```

В сценариях заголовки указываются в секции `headers`:

```
request:  
  method: GET  
  url: https://api.example.com/users  
  headers:  
    Authorization: Bearer ${token}  
    Accept: application/json
```

### Тайм-ауты

Тайм-аут запроса указывается в секундах:

```
zaplet get https://api.example.com/users -t 60
```

В сценариях:

```
request:  
  method: GET  
  url: https://api.example.com/users  
  timeout: 60
```

## Логирование

Zaplet поддерживает гибкое логирование с возможностью настройки.

### Настройка логирования

Конфигурация логирования находится в файле `config/logger.conf`:

```
[general]
name = zaplet
level = info
pattern = [%Y-%m-%d %H:%M:%S.%e] [%n] [%l] [thread %t] %v
async = false
async_queue_size = 8192
async_thread_count = 1

[console]
enabled = true

[file]
enabled = true
path = logs/zaplet.log
rotating = true
max_size = 10485760
max_files = 10
daily = false
rotation_hour = 0
rotation_minute = 0

[tcp]
enabled = false
host = 127.0.0.1
port = 9000

[udp]
enabled = false
host = 127.0.0.1
port = 9001
```

Доступные уровни логирования:

- **trace**: самый детальный уровень
- **debug**: отладочные сообщения
- **info**: информационные сообщения (по умолчанию)
- **warning**: предупреждения
- **error**: ошибки
- **fatal**: критические ошибки
- **off**: логирование отключено

## Устранение неполадок

### Распространенные проблемы и их решение

1. **Проблема**: Ошибка "Invalid URL"  
**Решение**: Убедитесь, что URL начинается с http:// или https://
2. **Проблема**: Ошибка "Connection refused"  
**Решение**: Проверьте доступность сервера и правильность URL

3. **Проблема:** Ошибка "Failed to parse file"

**Решение:** Проверьте синтаксис YAML в файле сценария

4. **Проблема:** Ошибка "Scenario file does not exist"

**Решение:** Проверьте путь к файлу сценария и его наличие

## Отладка и логирование

Для получения более подробной информации об ошибках измените уровень логирования на debug или trace в файле config/logger.conf.

---

**Примечание:** Это руководство покрывает основные возможности Zaplet. Для получения дополнительной информации об использовании HTTP-запросов и выполнении сценариев используйте команду `zaplet --help`. Подробная документация по созданию сценариев находится в файле `scenario_writing_ru.pdf`.