HTTP

- Common.HTTP.ProxyFetch — загрузка URL-контента через прокси

Common.HTTP.ProxyFetch — загрузка URL-контента через прокси

Пакет: `Http`
Полное имя компонента: `Common.Http.ProxyFetch`

Что это

ProxyFetch — системный PHP-компонент (V8Wrapper) для загрузки содержимого по HTTP/HTTPS URL внутри сценариев Metabot.

Проще всего воспринимать так:

это аналог `file_get_contents($url)`, но с поддержкой прокси, таймаутов и повторных попыток, на том же cURL-стеке, что используется для надёжных загрузок в платформе (`CdnFtp`).

Зачем нужен ProxyFetch

- `file_get_contents()` и часть HTTP-клиентов **не используют** прокси из окружения инстанса → запросы «уходят мимо» SOCKS5/HTTP-прокси и падают или «висят».
- Нет единых **таймаутов** и **ретраев** → сложнее переживать кратковременные сбои CDN и 5xx.
- Дублировать логику прокси в каждом плагине нецелесообразно.

ProxyFetch даёт **одну точку входа** для сценариев и плагинов: загрузить строку по URL или получить метаданные (HEAD) с теми же правилами отказоустойчивости.

Два контракта: generic fetch vs файловая загрузка

В платформе **два разных метода** с **разной семантикой успеха**. Не путайте их:

	`fetchUrlContents()` / **ProxyFetch**	`downloadFileFromUrlToBusiness()`
Назначение	Получить тело ответа в память	Скачать и сохранить файл на диск
`ok` / **успех**	`curl_exec !== false` и `http_code > 0`	Только `2xx + non-empty body`
404	`ok=true`, `http_code=404` — **by design**	Ошибка, return `null`
Пустой body	Допустим (204, HEAD-like API)	Ошибка
Retry	cURL error / `http_code=0` / `5xx`	cURL error / `http_code=0` / не-2xx / пустой body

> **Важно:** если кто-то «оптимизирует» один контракт под другой — сломает либо плагины, либо загрузку файлов.

Где используется

- Загрузка CSV/TXT/JSON по URL перед разбором в сценарии.
- Предварительная проверка размера/доступности файла по URL.
- Замена прямых `file_get_contents` в плагинах вроде `Common.Files.Converter`, `Common.Integrations.Request` (миграция по желанию команды).
- Любые кейсы, где бот на сервере обязан ходить в интернет **только через прокси**.

Где находится компонент

Подключение в JavaScript-сценарии (V8):

```
const ProxyFetch = require("Common.Http.ProxyFetch")
```

PHP-класс плагина:

`Plugins\Dynamic\Common\Http\V8Wrapper\ProxyFetch`

Платформенное ядро (можно вызывать из PHP-плагинов без V8):

`App\Services\CdnFtp::fetchUrlContents()`
`App\Services\CdnFtp::getFileInfoByUrl()` (расширенная сигнатура с прокси и ретраями)

Как работает ProxyFetch

Архитектура из трёх слоёв:

```
V8 JS  →  Common.Http.ProxyFetch  →  mergeWithDefaults (http_outbound + options)
       →  CdnFtp::fetchUrlContents / getFileInfoByUrl
       →  CdnFtp::curlExecWithFallback / curlExec  (общий cURL-раннер)
       →  cURL (прокси, таймауты, SSL, exec, диагностика)
```

Общий раннер: `curlExec()` — единая точка init/proxy/timeout/SSL/exec/close; `curlExecWithFallback()` — обёртка с `href_encode(url)` → fallback на raw URL. Переиспользуется в `fetchUrlContents` и `getFileInfoByUrl`.

Ретраи: отдельные «раунды»; в каждом раунде `curlExecWithFallback` перебирает варианты URL. Повтор, если `curl_exec` дал сбой, `http_code === 0` или `http_code >= 500`.

Логирование: при `HTTP_OUTBOUND_LOG_REQUESTS=true`:

- `CdnFtp outbound fetch: retry ...` / `failed after ...` — для fetchUrlContents
- `CdnFtp outbound fileInfo: retry ...` / `failed after ...` — для getFileInfoByUrl

Что нужно настроить

1. Конфиг `config/http_outbound.php`

Загружается автоматически; значения берутся из `.env`.

2. Переменные `.env`

Переменная	Назначение
`HTTP_OUTBOUND_PROXY`	Строка прокси (например `socks5://login:password@host:port`). Пусто — прокси не задаётся.

Переменная	Назначение
`HTTP_OUTBOUND_PROXY_VERSION`	Опционально: `v4` или `v6` (разрешение DNS/IP для cURL).
`HTTP_OUTBOUND_TIMEOUT`	Таймаут запроса, сек (по умолчанию 30).
`HTTP_OUTBOUND_CONNECT_TIMEOUT`	Таймаут на установку соединения, сек (по умолчанию 10).
`HTTP_OUTBOUND_RETRIES`	Число раундов попыток (по умолчанию 3).
`HTTP_OUTBOUND_RETRY_AFTER_SEC`	Пауза между раундами, сек (по умолчанию 2).
`HTTP_OUTBOUND_LOG_REQUESTS`	`true` / `false` — писать диагностические строки в лог (по умолчанию true).

Важно: эта группа **независима** от `botman.telegram`. Если на проекте один прокси на всё, можно задать одинаковые значения в обеих группах; если прокси разные — настраивайте раздельно.

3. Установка плагина на инстансе

Файл уже в `Plugins/Dynamic/Common/Http/V8Wrapper/ProxyFetch.php`. Автозагрузка через SPL autoload в `ModuleLoader` — дополнительных действий не требуется.

Создание плагина в админке

Если плагин **не** поставляется из репозитория, создайте его вручную:

1. **Плагин:** имя каталога/пакета `Http`, тип стандартный, **Common**, уровень доступа **SYSTEM**.
2. **Скрипт:** имя `ProxyFetch`, тип **PHP (WRAPPER FOR V8)**.
3. Вставьте код из файла `Plugins/Dynamic/Common/Http/V8Wrapper/ProxyFetch.php` (namespace и имя класса должны совпадать).
4. Включите скрипт (**is_enabled**).

Платформа сохранит PHP-файл в структуру динамических плагинов; `ModuleLoader` через SPL autoload видит класс автоматически.

Сигнатура вызова (JavaScript / V8)

`getContents`

```
const ProxyFetch = require("Common.Http.ProxyFetch")

let result = ProxyFetch.getContents("https://example.com/data.csv")

if (!result.ok) {
  debug("Ошибка загрузки: " + result.last_curl_error)
  return false
}

let text = result.content
```

С переопределением параметров:

```
let result = ProxyFetch.getContents("https://example.com/big.bin", {
  proxy: "socks5://user:pass@proxy.example:1080",
  timeout: 120,
  connect_timeout: 15,
  max_attempts: 5,
  retry_after_sec: 3
})
```

`getFileInfo`

```
let info = ProxyFetch.getFileInfo("https://example.com/photo.jpg")

if (info.exists && info.size_kb < 20480) {
  // условно безопасный размер
}
```

Параметры `options`

Поле	Тип	Обязателен	Описание
`proxy`	string	Нет	Переопределить прокси для этого вызова

Поле	Тип	Обязателен	Описание
`timeout`	number	Нет	Таймаут cURL, сек (дефолт из `http_outbound.timeout`)
`connect_timeout`	number	Нет	Таймаут соединения, сек
`max_attempts`	number	Нет	Число раундов ретраев (`http_outbound.retries`)
`retry_after_sec`	number	Нет	Пауза между раундами
`log_requests`	boolean	Нет	Логировать попытки (`http_outbound.log_requests`)

Формат ответа `getContents`

Поле	Тип	Описание
`ok`	boolean	Транспорт доставил ответ: `curl_exec !== false` и `http_code > 0`
`content`	string | null	Тело ответа; `null` при транспортном провале
`http_code`	number	HTTP-код последнего ответа
`attempts`	number	Сколько раундов было выполнено
`last_curl_error`	string	Текст ошибки при `ok === false`; при успехе — пустая строка

Про 404: при ответе 404 платформа вернёт `ok === true` и тело страницы ошибки — **всегда проверяйте** `http_code`, если вам нужен именно успешный ресурс. Это by design: generic fetch не навязывает семантику HTTP-статусов.

Формат ответа `getFileInfo`

Прежние поля:

- `exists` (0/1) — по сути «код ответа 200».
- `size_kb`, `type`, `mime_type`, `name`.

Добавлены:

- `http_code`

- `attempts`
- `last_curl_error`

Использование из PHP (без V8)

```
use App\Services\CdnFtp;

$result = CdnFtp::fetchUrlContents(
    $url,
    config('http_outbound.proxy'),
    (int) config('http_outbound.timeout', 30),
    (int) config('http_outbound.connect_timeout', 10),
    (int) config('http_outbound.retries', 3),
    (int) config('http_outbound.retry_after_sec', 2),
    (bool) config('http_outbound.log_requests', true)
);

if (!$result['ok']) {
    // логировать $result['last_curl_error']
}
```

HEAD / метаданные:

```
$info = CdnFtp::getFileInfoByUrl(
    $url,
    config('http_outbound.proxy'),
    (int) config('http_outbound.connect_timeout', 10),
    (int) config('http_outbound.timeout', 30),
    (int) config('http_outbound.retries', 3),
    (int) config('http_outbound.retry_after_sec', 2)
);
```

Одноаргументный вызов `CdnFtp::getFileInfoByUrl($url)` сохраняет старое поведение (без таймаутов из `http_outbound`).

Пример замены `file_get_contents` в плагине

Было:

```
$data = file_get_contents($url);
```

Стало (через ядро):

```
$fetch = \App\Services\CdnFtp::fetchUrlContents(
    $url,
    config('http_outbound.proxy'),
    (int) config('http_outbound.timeout', 30),
    (int) config('http_outbound.connect_timeout', 10),
    (int) config('http_outbound.retries', 3),
    (int) config('http_outbound.retry_after_sec', 2)
);
if (!$fetch['ok']) {
    throw new \RuntimeException($fetch['last_curl_error'] ?: 'fetch failed');
}
$data = $fetch['content'];
```

Обработка ошибок

Ситуация	Что ожидать
Таймаут	`ok === false`, в `last_curl_error` часто `cURL error 28: ...`
Прокси недоступен	`ok === false`, типично `cURL error 7: ...`
5xx после ретраев	`ok === false`, `last_curl_error` содержит `HTTP 503` и т.д.
404	`ok === true`, `http_code === 404` — проверяйте код
Невалидный URL / пустая строка	`ok === false`, `last_curl_error` поясняет причину

Сетевую ошибку и «битый» контент (HTML вместо CSV) различайте по **`http_code`** и валидации содержимого в сценарии.

Как отлаживать

1. `.env` — заданы ли `HTTP_OUTBOUND_PROXY`, таймауты и ретраи.
2. **Ручной** `curl` с того же сервера (с тем же прокси, если нужен) — доступен ли URL.
3. **Логи приложения** — строки `CdnFtp outbound fetch: ...` / `CdnFtp outbound fileInfo: ...` при включённом `HTTP_OUTBOUND_LOG_REQUESTS`.
4. `result.last_curl_error` / `info.last_curl_error` — конкретная причина отказа.
5. `http_code` — отличить 404/403 от обрыва соединения.
6. `attempts` — сколько раундов реально выполнено (ретраи работают).

FAQ

Можно ли задать другой прокси только для одного вызова?
Да, передайте `proxy` в объекте `options`.

Как отключить ретраи?
`max_attempts: 1` в `options` или `HTTP_OUTBOUND_RETRIES=1`.

Чем отличается от `bot.downloadFileFromUrl()`?
Скачивание в сценарий бота кладёт файл в хранилище бизнеса. ProxyFetch / `fetchUrlContents` возвращают **строку в памяти**, без сохранения на диск платформы. Кроме того, семантика успеха разная: download требует 2xx, fetch допускает 404.

Работает ли без прокси?
Да: если `HTTP_OUTBOUND_PROXY` пуст и в `options` не передан `proxy`, используется прямой cURL.

Нужен ли отдельный токен или callback?
Нет, это синхронный HTTP-запрос из PHP, без webhook'ов.

Что дальше

- Перевести `Common.Files.Converter` и `Common.Integrations.Request` на ProxyFetch / `CdnFtp::fetchUrlContents`.
- Перевести `downloadFileFromUrlToBusiness` на общий cURL-раннер (отдельная задача, follow-up).
- При необходимости расширить ядро (заголовки авторизации, POST) отдельной задачей.
- Держать в синхроне значения прокси с Telegram, если на проекте это один и тот же выход в интернет.

Связанные материалы

- Спека: `specs/Task-9703-proxy-fetch-plugin.plan.md`
- Код: `app/Services/CdnFtp.php`, `Plugins/Dynamic/Common/Http/V8Wrapper/ProxyFetch.php`, `config/http_outbound.php`