ASTRAVERGE / STORAGE

# Руководство пользователя

## AstraVerge Storage 1.4.0

Практическое руководство · 9 ноября 2026

Объектный доступ поверх вашей инфраструктуры.
Files · Pebble · PostgreSQL · AWS S3 / MinIO

Команды и имена файлов сохраняют техническое имя av-storage. Параметры и границы применимости относятся к версии 1.4.0.

# Содержание

1. Получите параметры подключения 3

2. Поймите адрес объекта 3

3. Запишите и прочитайте первый объект 3

4. Удалите тестовые данные 3

5. Используйте сетевой API 4

6. Правильно трактуйте успех 4

7. Действуйте при ошибке 4

8. Передайте запрос в поддержку 4

AstraVerge Storage 1.4.0 предоставляет приложениям единый доступ к бинарным объектам. Это руководство для инженера, который работает с уже настроенным сервисом. Установку, выдачу прав и выбор физического хранилища выполняет администратор.

# 1. Получите параметры подключения

Уточните у администратора версию клиента, локальный IPC endpoint или сетевой REST/gRPC endpoint, имя логического storage, разрешённый namespace и service token, если включена security. Передавайте token через защищённое окружение процесса; не включайте его в команды, историю shell или отчёт.

Для CLI используется AV_STORAGE_TOKEN. На Unix endpoint по умолчанию зависит от cwd: ./run/av-storage.sock. Явный -endpoint устраняет неоднозначность. На Windows используется named pipe. REST и gRPC должны быть включены администратором отдельно.

# 2. Поймите адрес объекта

Объект определяется тройкой (storage, namespace, key). Storage выбирает логический маршрут к files, Pebble, PostgreSQL или S3. Namespace разделяет данные приложения, key обозначает объект. Содержимое - произвольные bytes. Повторный PUT по тому же адресу заменяет значение.

Авторизация конечных пользователей остаётся у приложения. Namespace сам по себе не является правом доступа. Используйте только выданные storage и namespace; примеры ниже предполагают разрешённый тестовый namespace demo и storage files.

# 3. Запишите и прочитайте первый объект

Откройте PowerShell рядом с клиентом. Сервер должен быть уже запущен. Команды меняют только тестовый объект hello; убедитесь, что имя свободно.

```
.\av-storage-cli.exe ping
.\av-storage-cli.exe put files demo hello "HELLO"
.\av-storage-cli.exe get files demo hello
.\av-storage-cli.exe list-keys files demo
```

Ожидается OK, затем HELLO и список с hello. Успешный PUT не печатает подтверждение: проверьте $LASTEXITCODE сразу после команды, ожидается 0. Ping подтверждает доступность IPC, но не проверяет все backend или права на объекты.

На Linux/FreeBSD используйте ./av-storage-cli без .exe и проверяйте exit code средствами shell. Параметры токена и endpoint согласуйте с администратором.

# 4. Удалите тестовые данные

```
.\av-storage-cli.exe delete files demo hello
.\av-storage-cli.exe list-keys files demo
```

После DELETE проверьте exit code 0. В пустом demo ожидается []. GET удалённого объекта должен сообщить NotFound. Перед удалением рабочего объекта проверьте весь адрес; встроенной пользовательской корзины нет.

## 5. Используйте сетевой API

Пример для локальной доверенной проверки без security. Не открывайте такой listener наружу. Для production используйте выданный HTTPS endpoint, проверку CA/имени сервера и Bearer token через защищённый клиент.

```
curl.exe --fail-with-body -X PUT --data-binary "HELLO REST" http://127.0.0.1:8080/v1/files/demo/hello
curl.exe --fail-with-body http://127.0.0.1:8080/v1/files/demo/hello
curl.exe --fail-with-body -X DELETE http://127.0.0.1:8080/v1/files/demo/hello
```

LIST_KEYS, PUT_BATCH и GET_BATCH предоставляются локальным client IPC. В REST и gRPC этих методов нет. Для gRPC доступны потоковые Put/Get и унарный Delete. Полная матрица и правила кодирования адресов приведены в справочнике API.

## 6. Правильно трактуйте успех

При включённом WAL обычный PUT/DELETE подтверждает полную запись в журнал. Применение к физическому backend асинхронно. GET может прочитать pending значение из WAL: такое чтение не подтверждает завершение записи в PostgreSQL/S3. Fsync и batch ACK настраиваются отдельно; flush не означает fsync.

PUT_BATCH ram подтверждает volatile RAM queue: ещё не сброшенные записи могут потеряться при crash. Batch не является транзакцией; ошибка может оставить принятый префикс. Для GET_BATCH успех подтверждается терминальным END/OK, полным ожидаемым числом records и штатным EOF. Начало ответа не означает завершение всей операции.

## 7. Действуйте при ошибке

Timeout или разрыв связи оставляет результат неопределённым: запись уже могла быть принята. Отмена клиента не откатывает изменения. Не повторяйте запись вслепую при конкурентных writers: она может перезаписать более новое значение. Согласуйте стабильные ключи/версии и ограниченный retry с backoff в приложении; CAS и idempotency key API отсутствуют.

При 401 проверьте наличие credential в нужном процессе; при 403 - выданные права, включая отдельное list. При 503/Unavailable обратитесь к администратору с временем и адресом операции. Не выводите token, содержимое объектов или DSN в общий журнал. Не исправляйте сбой удалением WAL.

## 8. Передайте запрос в поддержку

Соберите версию клиента/сервера, ОС, время, endpoint без секретов, последовательность команд и exit codes, статус полного/частичного ответа. Администратор добавит обезличенный inspect-config, status и stderr. WAL содержит значения объектов и передаётся только по согласованному защищённому каналу.

Продолжение: установка, настройка, безопасность, администрирование, диагностика.