

Архивные операции

Раздел **Архивация** создаёт архивы из текущего Workbench **Источник** в текущую **Назначение**. Он использует portable runtime и внешние backend-инструменты, если они доступны.

Форматы

Формат	Выход	Комментарий
zip	.zip	Универсальный формат, открывается везде. Шифрование недоступно, см. ниже.
7z	.7z	Хорошее сжатие. Единственный формат с шифрованием: AES-256, с именами файлов или без.
sfx	.exe	7Z self-extracting executable.
lz4	.tar.lz4	Дополнительный TAR плюс LZ4 через lz4.exe . Показывается только если backend найден.
tar	.tar	Без сжатия или как контейнер.
gz	.tar.gz	TAR плюс gzip.
zstd	.tar.zst	TAR плюс zstd, нужен backend.

Недоступные backend-ы не должны запускаться вслепую. GUI показывает только доступные/валидные варианты или даёт disabled reason. В обычной portable-сборке основная линейка форматов выглядит как ZIP, 7Z, SFX, TAR, TAR.GZ и TAR.ZSTD; LZ4 появляется только при найденном backend.

LZ4 идёт через отдельный **lz4.exe**, а не через 7-Zip: во вложенной сборке 7-Zip lz4 присутствует только как кодек, поэтому **7z a -tlz4** возвращает **Unsupported archive type**. Backend ставится кнопкой **INSTALL LZ4** в панели **Архивация** или шагом **[06] LZ4** в **builder_main.cmd**; оба вызывают **install\Install-Portable-LZ4.cmd** и раскрывают **lz4.exe** в **Tools\lz4**.

Правовая заметка: CLI `lz4.exe` распространяется под `GPL-2.0-or-later`, сама библиотека liblz4 - под `BSD-2-Clause`. В upstream-пакете `lz4_win64_*.zip` отдельного файла LICENSE нет, поэтому перед релизом lz4 нужно зарегистрировать в `Audion License Fleet Manager` вместе с остальными компонентами.

Уровни сжатия

- `0` - быстро, почти без сжатия.
- `1` - дефолт для большинства рабочих задач.
- `3`, `5`, `7` - компромисс скорость/размер.
- `9` - максимальное сжатие, может быть медленным.

Для network archive доступны только `0`, `1`, `3`, `5`. В GUI уровень сжатия вынесен в компактный spinner слева от линейки форматов.

Шифрование

Режим выбирается кнопками, а не выпадающим списком: `Не шифровать`, `Зашифровать`, `Зашифровать с именами`. Под кнопками GUI называет алгоритм и форматы, которым режим доступен.

- `none` - без шифрования.
- `password` - шифровать содержимое. Имена файлов внутри архива остаются видимыми.
- `names` - шифровать и содержимое, и имена файлов.

Оба режима шифрования доступны только для **7Z и SFX** и всегда дают **AES-256** (`7zAES` с 2^{19} итерациями KDF). SFX собирается как 7z, поэтому получает то же самое.

ZIP не шифруется намеренно. 7-Zip закрывает ZIP устаревшим ZipCrypto - шифром PKZIP, вскрываемым атакой по известному открытому тексту, а заголовки внутри ZIP предсказуемы. Альтернатива `-mem=AES256` даёт стойкий архив, но такой ZIP не открывается встроенным Проводником Windows, и ZIP перестаёт быть форматом «откроют везде». Поэтому ZIP оставлен без шифрования, а для защищённого архива берите 7Z или SFX.

TAR, TAR.GZ, TAR.ZSTD и TAR.LZ4 шифрования не имеют вовсе: в самих форматах его нет, а ZSTD и LZ4 вдобавок собираются отдельными `zstd.exe` и `lz4.exe`.

При включённом шифровании неподдерживаемые форматы отключаются в списке и объясняют причину в подсказке, а уже выбранные - снимаются. Кроме того,

`assert_encryption_supported` падает до сборки архива, поэтому «зашифрованный TAR» не может появиться даже в обход GUI.

Пароль вводится в маскированное поле. GUI log редактирует аргументы архивации вида `-p...` как `-p[REDACTED]`, чтобы секрет не попадал в текущий log. CLI-архиваторы всё равно получают пароль как аргумент процесса, поэтому не вводите его в правый terminal command bar и не запускайте приватные архивы на чужой машине.

Имена файлов

`archive_name_prefix` добавляется перед именем исходного элемента.

`archive_name_suffix` добавляется после имени исходного элемента перед расширением.

Последовательная нумерация:

- `N.` или `N_` в prefix включает numbering перед именем;
- `_N` в suffix включает numbering после имени.

Запрещённые Windows-символы в имени очищаются.

Layout

`flat` - в GUI подписан как `ЦЕЛЬ`: архивы пишутся прямо в `TARGET`.

`per_folder` - в GUI подписан как `ЦЕЛЬ/<имя>`: каждый исходный элемент получает подпапку `TARGET\<name>`.

SFX auto extract

`sfx_extract_path` задаёт путь установки/распаковки для SFX.

`{name}` заменяется именем исходного элемента.

Env selector может использовать известные Windows env roots, например `%USERPROFILE%`, `%PROGRAMDATA%`, `%TEMP%`, `%DESKTOP%`.

Пустой путь оставляет default behavior SFX.

SFX wrappers

Если выбран `sfx`, можно дополнительно создать wrapper archive: ZIP, 7Z, ZSTD, TAR, GZ. Это удобно, когда `.exe` нельзя отправить напрямую или нужно сделать транспортный контейнер.

Wrapper создаётся после успешного SFX.

Проверка и удаление исходников

`archive_verify_after_create` проверяет каждый созданный архив сразу после сжатия.

ZIP/7Z/TAR/GZ проверяются через `7z t`; ZSTD-потоки, включая `.tar.zst`, проверяются через `zstd -t`.

`archive_delete_source` удаляет исходные файлы и папки только после успешного создания выбранных архивов и успешной проверки, если она включена. Используйте осторожно: это workflow "упаковать, проверить и убрать исходник".

Для важных данных сначала сделайте тест без удаления.

Практические сценарии

Для быстрой передачи большого дерева мелких файлов по сети часто лучше:

1. Создать `7z` с уровнем `1`.
2. Включить integrity test.
3. Передать один archive через network transfer.
4. Проверить SHA256 на принимающей стороне.

Для долгого хранения:

1. Использовать `7z`.
2. Уровень `5` или `7`.
3. Включить password/encrypt names, если нужны приватность и защита имени файлов.
4. Сохранить пароль вне проекта.