

Audion TC — устройство сборки

Портативная сборка Total Commander 11 x64. Файл описывает решения, которые не видны из кода за минуту: почему сделано так, а не иначе.

Главное правило: в корне ничего своего

Обновление Total Commander распаковывается **поверх** папки программы и затирает всё, что совпало по имени. Поэтому:

- код — скрипты, карта программ, `bin\fzf.exe` — живёт в `Scripts\`;
- запускалки — в `Run\`;
- меню и языковой файл — в `Menus\`, а не в `Language\` (это территория TC);
- стартер `Start Audion TC.cmd` лежит в `Run\` (мастер-копия) и продублирован в корень для удобства. Он сам определяет, где запущен, по наличию рядом `TOTALCMD64.EXE`, поэтому обе копии побайтово одинаковы. Снесло корневую — пульт вернёт её пунктом «Восстановить стартер в корне».

Код и запускалки врозь

`Scripts\` — только `.ps1` и данные. `Run\` — только `.cmd`, то, что человек или Total Commander запускает: `Launcher.cmd`, `ps.cmd`, `New-File.cmd`, `Update-All.cmd` и прочие. Раньше они лежали вперемешку, и в списке из двадцати семи файлов приходилось выяснять, что тут точка входа, а что её внутренности.

Врапперы зовут код этажом выше: `call "%~dp0ps.cmd" "%~dp0..\Scripts\X.ps1"`. `ps.cmd` и `psadmin.cmd` остаются соседями остальных запусколок — они сами запускалки, выбирающие интерпретатор.

Пути в `Usercmd.ini` и в генераторе панелей ведут в `Run\`: `%COMMANDER_PATH%\Run\psc.cmd`. Поменять раскладку — значит переписать и их.

Кнопки и меню собираются сами

`Scripts\apps_map.ini` — карта: одна секция = одна кнопка. Группы:

Группа	Куда попадает
<code>main</code>	начало блока в главной панели (TeraCopy Copy/Move)
<code>tail</code>	конец блока (Zapret, TG WS Proxy)
<code>system</code> / <code>admin</code> / <code>media</code>	одноимённые панели <code>Bars*.bar</code>

`Scripts\Sync-Apps.ps1` проверяет каждый файл на диске и переписывает панели, а также блоки в `Usercmd.ini` и `Menus\WCMD_AUDION_RUS.MNU` между маркерами `; >>> AUDION APPS` ... / `; <<< ...`. Снаружи маркеров — ручная территория.

Своими генератор считает кнопки, ведущие в `Apps\`, в `Run\Launcher.cmd` или в любую `Bars\*.bar` кроме `audion.bar`, `Vertical.bar`, `No.bar`. Список именно по исключению: при перечислении переименование группы оставляло в панели кнопку на исчезнувший файл.

Унесли `Apps` целиком — панели соберутся пустыми, ссылки на них исчезнут, менеджер останется рабочим и займёт 171 МБ вместо 1,8 ГБ.

Консольные программы — через обёртку. Кнопка обычно ведёт прямо на `.exe`, но программа, которая печатает и сразу выходит (fastfetch), в своём окне только мелькнула бы. Для таких в карте есть `run=<обёртка.cmd>`: кнопка ведёт на неё (`Run\Fastfetch.cmd` запускает fastfetch с портативным конфигом `Ware\fastfetch.jsonc` и держит окно `pause`), а значок при этом остаётся от самой программы — `button=` смотрит на `.exe`, а на обёртку только `cmd=`. Конфиг лежит в `Ware`, а не в `Apps\fastfetch`: обновление подменяет папку в `Apps` целиком и унесло бы его. Кнопка появляется, только когда fastfetch есть в `Apps` (как у всех), — в облегчённом релизе и пакете-наложении её нет, пока программу не поставят.

Сам fastfetch — визитка системы (разовый бриф о железе и ОС), а не монитор нагрузки; что он показывает и как править конфиг — в `Docs\fastfetch.md`.

Так же устроен **bottom** (btm) — живой монитор нагрузки: та же `run=`-обёртка (`Run\Bottom.cmd`), только окно она не держит `pause`, а отдаёт самому bottom — он интерактивный, выход по `q`. Конфиг — `Ware\bottom.toml`, передаётся ключом `-C` (в 0.14 это `--config_location`, не `--config`); подробности в `Docs\bottom.md`. fastfetch и bottom **предустановлены как резиденты** — в облегчённом релизе и в пакете-наложении (`Apps`, ~17

и ~5 МБ), чтобы визитка и монитор были под рукой без установки; остальное каталог доставляет из пульта.

Справка: лежит в корне, обновляется из e\

В `INSTALL.CAB` справка разложена по языкам: `e\TOTALCMD.CHM` и `e\KEYBOARD.TXT` (английские), `d\TOTALCMD.CHM` и `d\TASTEN.TXT` (немецкие). Мастер установки копирует нужную пару в корень, а обе папки удаляет — сборка выглядит так же.

Из-за этого справка однажды застряла. Обновление копирует только то, что в сборке **уже есть по тому же пути**: файл в корне против файла в `e\` — пути не совпадают, и справка молча осталась от 11.56, пока сам менеджер доехал до 11.58.

Лечится не переносом справки в `e\` (папка в корне сборки выглядит как забытый мусор установщика), а картой соответствий `$moveToRoot` в `Update-TotalCommander.ps1`:

```
'e\TOTALCMD.CHM' = 'TOTALCMD.CHM'
'e\KEYBOARD.TXT' = 'KEYBOARD.TXT'
```

Обновление берёт файл по пути дистрибутива и кладёт туда, где его держим мы. `KEYBOARD.TXT` иначе и нельзя: его Total Commander ищет только в корне, ключа для него нет. У справки ключ есть — `HelpFile`, — но и она в корне, чтобы раскладка сборки не отличалась от обычной установки.

Немецкая справка уезжает в `Ware\Stock` вместе с папкой `d\`.

Проверять просто: подменить оба файла заглушкой и запустить обновление — оно должно вернуть их из дистрибутива.

Плагины обновляются той же картой

Плагины Total Commander лежат в `apps_map.ini` группой `plugin` — рядом с программами, но без кнопок: группа `plugin` в панели не попадает. Вместо `dir` у них `target`, потому что живут они не в `Apps`, а в самой сборке (`plugins\WLX64\...`). Файл в `exe` — это `.wlx` / `.wdx` / `.wfx`: по нему загрузчик находит корень внутри архива.

В архивах обе разрядности лежат рядом (`X.wlx` и `X.wlx64`) — менеджер сам берёт нужную, поэтому кладем как есть.

Единого каталога у плагинов нет, источники разные:

- `plugins.ghisler.com/content/wdx_*.zip` — постоянный адрес без версии в имени, но там не всегда свежее: у DirSizeCalc лежала 2.21 против 2.22 у автора;
- `totalcmd.net/download.php?id=<имя страницы plugring>` — тоже постоянный адрес, редирект на сайт автора. Саму страницу `plugring/<id>.html` разбирать не нужно и не стоит: ссылка на ней записана как раз через `download.php`;
- `github` — у тех, кого продолжают разрабатывать (xPDFSearch).

Версию установленного плагина пишет `pluginst.inf` рядом с ним. Часть авторов это поле не заполняет — тогда версия берётся из самого `.wlx` / `.wdx`.

Регистрация плагинов — отдельная история, не через `overlay.ini`: просмотрщики и колонки нумеруются строками (`0=`, `1=`), поэтому слияние по именам ключей вытеснило бы чужие плагины молча. Установка пишет свои записи только в пустую секцию, иначе не трогает и говорит об этом. Файловые плагины (`WebDAV`) устроены иначе — там ключ это имя, под которым плагин виден в сетевом окружении.

Версии в именах файлов

`GPU-Z.2.70.0.exe`, папка `Xvid4PSP 8.2` — версия меняется при каждом обновлении. В карте пишутся маски (`exe=GPU-Z*.exe`, `dir=rufus*`), генератор берёт самое свежее совпадение и подставляет в кнопку реальное имя.

Загрузчик

`Scripts\Get-App.ps1` — скачивает портативные сборки прямо в `Apps\<папка>`. Шесть источников:

<code>source=</code>	Как берёт
<code>techpowerup</code>	три запроса за подписанной ссылкой; вариант выбирается ключами <code>pick</code> (маска имени) и <code>kind</code> (<code>ZIP*</code> , <code>Portable</code>)
<code>winget</code>	<code>winget download --installer-type zip portable</code> ; вывод не глушится, WinGet рисует свой прогресс
<code>github</code>	последний релиз, ассет по маске
<code>sourceforge</code>	лента файлов проекта (<code>rss?path=</code>), файл по маске; свежие в ленте сверху

<code>source=</code>	Как берёт
<code>webpage</code>	ссылка ищется на странице по маске; <code>linkdeny</code> отсекает лишнее (preview-сборки)
<code>url</code>	постоянный адрес с редиректом на свежую версию (так раздаёт себя VS Code); имя файла берётся из того, куда привёл редирект

Только стабильные выпуски. `/releases/latest` на GitHub сам пропускает prerelease, поэтому Windows Terminal берётся стабильный, а не Preview. Где беты лежат рядом со стабильными, их отсекает маска: у HWiNFO бету выдаёт второе подчёркивание (`hwi_851_6045.zip` против `hwi_850.zip`), у CrystalDiskInfo и CrystalDiskMark обычную редакцию от Shizuku, Aoi, Ads и Src отличает цифра в конце имени.

Несколько подходящих файлов. Проект может собрать сразу пару веток — FFmpeg кладёт в один релиз сборки из `master`, `n7.1` и `n8.1`. Берётся самая свежая, и числа сравниваются как числа: по алфавиту `n10` оказался бы старше `n8`.

Лимит GitHub API — 60 запросов в час на анонимные вызовы, и он выбирается за одну разведку. Поэтому `github` при ответе `rate limit exceeded` молча переходит на HTML: `https://github.com/<repo>/releases/latest` редиректит на тег, а `.../releases/expanded_assets/<tag>` отдаёт полный список файлов со ссылками. Квот там нет. Проверено на живом обновлении Notepad++ при исчерпанном API.

Разрядность. GitHub кладёт arm64 и x64 рядом, и маска вида `*bin.zip` цепляет arm64 первым. В масках разрядность указывается явно: `systeminformer-build-win64-bin.zip`, `ImageGlass*_x64.zip`.

Не всё живёт в Apps. Ключ `target` задаёт путь относительно корня сборки — так качаются консольные редакторы micro и MS Edit в `Tools\` и плагины TC, не заводя второй копии. GUI-редакторы переехали в `Apps\` (`dir=`), как обычные программы.

Прогресс. `Invoke-WebRequest -OutFile` не показывает ход загрузки вовсе, и на большом файле окно выглядит зависшим. Загрузка идёт потоком через `HttpClient` с обновлением строки процентов раз в 300 мс.

Когда сайт не пускает .NET. hwiinfo.com отвечает 403 и на страницу, и на файл, если запрос пришёл из PowerShell: фильтр узнаёт .NET по рукопожатию TLS, и подделка заголовков не помогает — браузерный `User-Agent`, `Accept`, `Referer` дают тот же 403. Штатный `curl.exe` из Windows тот же запрос проводит без вопросов, поэтому при 403 загрузчик молча повторяет попытку через него.

Страница вместо файла. SourceForge браузерному агенту показывает «загрузка начнётся через несколько секунд», а простому загрузчику сразу отдаёт файл — оттуда качаем под `Wget/1.21.4`. На такой случай есть общая проверка: если вместо архива пришёл `text/html`, загрузчик говорит об этом прямо, а не падает позже на распаковке.

Порядок работы: скачать во временную папку → распаковать → **убедиться, что внутри есть файл по маске из карты** → только тогда подменить папку. Препятствие остаётся рядом как `.bak` (убирается пунктом пульта). Ключ `keep` переносит нажитое программой — у FastStone это `FSSettings.db`, у Zapret `lists\*-user.txt`: маска умеет указывать и в подпапку.

Освободить файлы перед заменой. Ключ `prerun` вызывает скрипт из `Scripts` ровно в тот момент, когда всё уже скачано и проверено, но папка ещё не тронута. Так сделан Zapret: `prerun=Update-Zapret.ps1 -StopOnly` снимает драйвер WinDivert, который держит файлы. Момент выбран не случайно — сними драйвер раньше, и за новой версией пришлось бы идти на GitHub уже без обхода блокировок.

Когда прежняя версия не нужна. `nobackup=1` меняет саму механику замены: папка не подменяется снаружи, а чистится изнутри и наполняется заново. Копий не остаётся, и замена проходит даже тогда, когда папку держит открытая панель ТС — переименовать её в такой момент нельзя, а вычистить можно. Стоит у Zapret: папка с этим именем принадлежит ему целиком, что бы в ней ни лежало — пустая, недокачанная или обкусанная антивирусом (Defender сносит `bin\winws.exe` как «HackTool»). Разбираться, что испортилось, незачем — свежий выпуск чинит любой случай. Уцелевает только то, что указано в `keep`.

Остальным `.bak` остаётся страховкой, но молчаливой она быть не должна: после каждого обновления загрузчик пишет, сколько её накопилось, а в пульте пункт показывает объём прямо в подписи — «Очистить *.bak после обновлений (10 шт, 95 МБ)». Иначе за полгода сборка тихо растёт вдвое.

`.bak` **ложится рядом с программой, а не в Apps**. Копия остаётся там, где живёт сама папка, и у записи с `target` это `Ware\npp.bak` или `plugins\WDX64\Exif.bak`. Пока и очистка, и подпись пункта смотрели только в `Apps`, таких копий не видел никто: пульт честно писал «(пусто)», а `Ware\npp.bak` на 10 МБ лежал в сборке с самого обновления и не убирался ничем. Теперь места собираются по той же карте — `Apps` плюс папка над каждым `target`, — и очистка со счётчиком считают их все; в строке «убрано» стоит путь от корня сборки, иначе по имени `npp.bak` не понять, откуда оно.

Самораспаковывающиеся `.exe` (DDU, PeaZip, foobar2000) вскрываются через 7-Zip из `Apps\peazip`, с откатом на системный. После NSIS убирается служебный сор (`$PLUGINS\DIR`, `$R0`).

Значки в главном меню

Устроено не так, как кажется, и на этом потеряно много времени. Значок пункта меню задаётся **таблицей соответствий**, а не рядом с командой:

Файл	Что делает
<code>Wcmicon2.dll</code>	библиотека значков (у нас расширенная, 1615 штук)
<code>wcmicon2.inc</code>	таблица «команда → номер значка в этой библиотеке»

Файл таблицы выбирается по имени библиотеки: при `DefaultLib=wcmicon2` читается `wcmicon2.inc`, и только если его нет — штатный `WCMICONS.INC`. Именно так сообщество раздало расширенный набор штатным пунктам меню: 512 строк вида `69=0`, где слева номер внутренней команды Total Commander.

Наши команды `em_audion_*` прописываются в тот же файл номерами **с 10000, по порядку следования в меню** — не в `Usercmd.ini`, а именно в меню: первая встреченная сверху вниз получает 10000, вторая 10001 и так далее. Поэтому таблицу нельзя править отдельно от меню: переставили пункт — пересоберите `Scripts\Build-MenuIcons.ps1`.

Что **не** работает, хотя выглядит логично:

- `button=` в `Usercmd.ini` — только для кнопок панели, меню его не читает;
- `wciconex.dll` с `wciconex.inc` — игнорируется, когда активна `wcmicon2`;
- ссылка на любую другую библиотеку** — с версии 11.50 меню принимает значки только из `wcmicons.dll`, отдельных `.ico` или самого `totalcmd.exe`.

Проверять правки удобно, не трогая эталон:

```
"TOTALCMD64.EXE" /i="Ware\State\Test.ini"
```

Ключ `/i=` уводит запись настроек во временный файл, и `Wincmd.ini` остаётся нетронутым — иначе менеджер при выходе переписет в нём размеры окна под текущий монитор.

Перебивка штатного меню на значки Windows

Штатные пункты Гислера (файловые операции, навигация, вид) шли на наборе сообщества эпохи Windows 7 — на нынешних экранах он выглядит грубо. Мы ставим им современные значки Windows по смыслу команды.

- `Ware\Templates\stock-icons-plan.ini` — план: `<команда TC> = shell32:N` (или `imageres:N`, или `mirror:shell32:N:h` для зеркальных пар назад/вперёд);
- `Scripts\Build-StockIcons.ps1` — кладёт эти значки в `Wcmicon2.dll` и пишет `Ware\Templates\stock-icons.ini` (готовые `команда = номер в библиотеке`);
- `Build-MenuIcons` накладывает `stock-icons.ini` поверх унаследованной таблицы.

Всё видимое меню (280 пунктов) переведено. В таблице ещё остаются строки на старый набор — это команды TC вне меню (диалоги, список команд); их оставили как есть. Индексы из тусклой зоны (`shell32` 279+) не берём — только цветные.

Значки кнопок панелей

Кнопки главной панели держат значки в общей `Wcmicon2.dll` — той же библиотеке, что и меню, поэтому стиль единый.

- `Ware\Templates\bar-icons-plan.ini` — план `глиф = источник`. Источники: `shell32:N` (единый стиль с меню), `prog:calc.exe` / `prog:mstsc.exe` (значок самой программы — для Калькулятора и RDP), `png:terminal` (логотип Windows Terminal из `Ware\Icons\terminal.png`). Часть глифов (стрелки навигации, глаз скрытых, инвертирование, файл/папка с плюсом, вкладки) лежит в `Wcmicon2.dll` с давних пор — отдельного источника им не нужно, их держит карта слотов;
- `Ware\State\bar-slots.ini` — карта «глиф → слот в `Wcmicon2.dll`». По ней `Build-BarIcons.ps1` переиспользует готовые значки в любой сборке (Base, Release, Ultimate), а не кладёт дубли;
- `Scripts\Build-BarIcons.ps1` — добавляет недостающие значки в `Wcmicon2.dll` и переписывает `button=` в `Bars\audion.bar` / `Bars\Vertical.bar` на нашу библиотеку. `Sync-Apps` при пересборке `audion.bar` ручные кнопки не трогает, поэтому наши `button=` на `Wcmicon2.dll` держатся.

Размеры значков на разных масштабах

Значок — не картинка, а набор картинок фиксированных размеров: 16, 24, 32, 48, 64. Когда ТС просит размер, которого в наборе нет, Windows берёт ближайший и растягивает с дробным коэффициентом: линия толщиной в один пиксель становится толщиной 1,16 и попадает между пикселями. Отсюда рёбра и муть.

Кратность восьми сама по себе ни при чём: 40 кратно, но значков в 40 пикселей почти никто не рисует — результат тот же. Зато 16 → 32 и 24 → 48 (ровно вдвое) масштабируются без потерь даже когда точного размера нет.

Поэтому лестница в `Apply-Config.ps1` не считается пропорцией, а перечисляет только настоящие размеры:

Масштаб	dpi	Главная	Боковая	Меню
100%	96	32/32	16/16	16
125%	120	32/32	16/16	16
150%	144	32/32	16/16	16
175%	168	24/24	16/16	16
200%	192	24/24	16/16	16
250%	240	24/24	16/16	16

Размер убывает с ростом масштаба, и это не ошибка. Windows уже растянула окно, поэтому значку в 32 пикселя при 200% досталось бы вдвое больше места, чем при 100%, и панель разрослась бы в полосу. 32 внизу и 24 наверху дают на глаз одну и ту же кнопку.

Высота кнопки равна размеру значка: поля ТС добавляет сам. Боковая панель и меню от масштаба не зависят вовсе — там всегда 16: это списки, а не панель запуска, и мелкий значок позволяет уместить в столбец больше.

Две пары ключей на одну высоту. Кроме лестницы `Buttonheight<dpi>` ТС держит `Buttonheight` и `SmallIconSize` без суффикса — значения для `DefaultDpi`, от которых он отсчитывает высоту, когда точного ключа под текущий экран не нашлось. На этом сборка и попалась: от эталона остался `Buttonheight=37` при `DefaultDpi=144`, хотя в лестнице на 144 стоит 32. Выглядело странно — значки на всех экранах стояли твёрдо, а высота кнопок плавала. Разгадка в том, что `SmallIconSize=32` с лестницей случайно совпал, а `Buttonheight=37` —

нет. Теперь `Apply-Config.ps1` приводит обе пары к той строке лестницы, на которую указывает `DefaultDpi`.

Размер окна: `ResolutionSpecific=0` плюс строки `monitor(...)`. Ключ пробовали в положении `1` — тогда ТС заводит секцию вида `[2560x1440 (8x16)]` под каждый экран. Мысль здравая, но ломается на первой встрече: на незнакомом экране ТС берёт не эталон из `[AllResolutions]`, а собственные умолчания, и колонки с окном встают чужие. Настроенная раскладка при этом цела — просто не применяется.

Оказалось, что посекционный режим и не нужен: раскладку по мониторам ТС помнит и при `0`, отдельными строками в `[AllResolutions]`:

```
monitor(0,0,1920,1200;144)=17,68,1878,1043    ноутбук
monitor(0,0,2560,1440;144)=242,57,2089,1253    27 дюймов
monitor(0,0,3840,2160;144)=732,377,2226,1337    4К
```

Слева — границы монитора и dpi, справа — позиция и размер окна. Строку ТС дописывает сам, когда на этом экране выполнили **Конфигурация → Сохранить позицию** (просто выйти из ТС недостаточно: положение окна он по выходу не сохраняет).

А `x/y/dx/dy` и `Tabstops` в той же секции — запасной вариант для экрана, который ТС видит впервые. Считается он не по нашим экранам, а по худшему из ходовых: **1920×1080**. Таких мониторов больше всего, и раскладка с ноутбучных 1200 строк в них не влезает — 68 сверху плюс 1043 высоты дают 1111 при доступных примерно 1020. Отсюда базовые `17,20,1878,950` и ширины колонок `249,257,705,179,683,93`:

экран	рабочая высота	запас снизу
1920×1080 @100%	1032	62
1920×1080 @125%	1020	50
1920×1080 @150%	1008	38
1920×1200 @125%	1140	170

Рассчитанное на большой экран сюда не годится вовсе: окно шириной 2226 в 1920 не помещается и уезжает за правый нижний угол. Обратная ошибка безобидна — на экране покрупнее просто останется поле, а дальше сработает строка `monitor(...)`.

Ловушки, на которых уже спотыкались

- ТС разворачивает `%COMMANDER_PATH%` в поле команды, но **не в поле параметров**. Поэтому путь к скрипту не передаётся параметром — обёртка (`New-File.cmd`) находит его рядом с собой.
- `%P` приходит с завершающим слешем и съедает закрывающую кавычку. В командах пишется `"%P."`, а скрипт нормализует путь для показа.
- `shift` в `.cmd` сдвигает и `%0` — папку скрипта надо запомнить до сдвига.
- Кириллица в `.cmd` ломается: консоль в cp866, файлы в UTF-8. Комментарии и сообщения в `.cmd` — по-английски.
- WinForms-диалог мылит на HiDPI, пока процесс не объявил себя осведомлённым: `SetProcessDpiAwarenessContext(-4)` до создания первой формы.
- У упорядоченного словаря PowerShell числовой ключ читается как номер позиции. Таблицы «dpi → размер» держим списками.
- `Wincmd.ini` ТС пишет то с CRLF, то с одиночным LF — разделитель определяется по файлу и возвращается тот же.
- ТС перечитывает `Usercmd.ini` и панели только при старте, а `Wincmd.ini` переписывает при выходе (и способен затереть горячие клавиши).

Чужое лежит на своих местах

Правило, к которому пришли: файлы Гислера остаются оригинальными и там, где их кладёт дистрибутив, а конфиг просто ссылается на них. Ничего чужого мы не переименовываем и не правим.

Как было и стало:

Было	Стало
<code>Menus\WCMD_AUDION_RUS.LNG</code> — копия штатного языкового файла	<code>Language\WCMD_RUS.LNG</code> , ссылка в <code>LanguageIni</code>
<code>Menus\WCMD_AUDION_RUS.INC</code> — копия штатного, переименованная	<code>Language\WCMD_RUS.INC</code>
меню <code>WCMD_AUDION_PETERMAD_*</code>	убраны: наше меню давно переписано, чужое не нужно

Копия языкового файла отличалась от оригинала ровно одной строкой — названием языка «Russian (Audion)». Ради этого держать производное не стоило.

Своим остаётся то, что своим и является: `WCMD_AUDION_RUS.MNU` с `.INI` — наше меню, панели, цвета, скрипты. Справка `TOTALCMD.CHM` в корне — русская, в официальный дистрибутив она не входит и в раздачу не пойдёт.

Просмотрщик: почему без CudaLister

CudaLister снят с регистрации, а потом и удалён из `plugins\WLX64` — 11,5 МБ лежали без дела. Причина видна в конфиге: у него, единственного из шести, **не было ключа** `detect` — то есть он перехватывал вообще все файлы, а не свой тип. Отсюда и падения ТС на файлах с кодом. Остальные просмотрщики ограничены типами и работают.

Смотрит теперь штатный Lister: тёмный режим у него включается общим `DarkMode=2`, цвета берутся из секции `[Lister]`. Подсветки синтаксиса нет — для кода F4 открывает выбранный редактор (по умолчанию Notepad, см. ниже).

Редакторы: F4-переключатель на шесть

Клавишу F4 (правка файла под курсором) обслуживает `Editor=` в `[Configuration]` `Wincmd.ini`. По умолчанию там системный **Notepad** — знакомый всем, с вкладками, и он следует теме Windows (светлая тема ТС → светлый Notepad). Для не-гиков этого хватает; кто хочет другой — переключает в меню «Audion TC». `Set-Editor.ps1` (враппер `Run\Set-Editor.cmd`, команды `em_audion_editor_*`) правит `Editor=` как темы — на закрытом ТС, с перезапуском; активный отмечен bullet-маркером в тексте пункта (ТС не даёт колонку-галочку пользовательским командам).

Где живут и динамика меню. Пять GUI-редакторов — в `Apps\` (`apps_map.ini`, группа `media`, `dir=...`), качаются и обновляются пультом как любая программа. Textadept и Lite XL едут в сборке настроенными; CudaText приезжает с русским интерфейсом и тёмной темой ebony, Notepad3 — с тёмной темой Sombra: обе настройки кладёт заготовка `Profiles\<prog>` при установке (у CudaText это `settings\user.json`, у Notepad3 — `Notepad3.ini`). Notepad++ — особая статья (GPL, см. ниже): в сборке его нет, ставится отдельной кнопкой. Пункт переключения в меню «Audion TC» `Sync-Apps.ps1` **собирает по факту наличия**: удалил папку редактора (руками или пультом) — пункт исчезает; докачал — возвращается (Notepad системный, есть всегда, поэтому в списке постоянно). Архив с папкой-обёрткой внутри (`textadept\`, `lite-xl\`) загрузчик разворачивает сам: `payloadRoot` — папка найденного

exe, поднятая на глубину подпапки из `exe=`. Консольные редакторы держим отдельно, в `Tools\`: micro (обёртка `Run\Micro.cmd`) и MS Edit (`Tools\msedit`, `nopath=1`, `nobackup=1`, по `Alt+E`); в F4-переключатель они не входят.

Кодировки — линия раздела. Textadept и Notepad едят любой файл, включая cp1251-меню и UTF-16 `.bar`. Lite XL — строго UTF-8 (на cp1251 его токенайзер падает на `ulen`), поэтому он для UTF-8-зоны: MD, JSON, YAML, код. MS Edit UTF-16 LE BOM ещё портит (баг #545/#752) — на `.bar` не звать.

Notepad++ — единственный, кого нет в коробке. Остальные редакторы пермиссивно-лицензированы (Textadept и Lite XL — MIT, Notepad3 — BSD-3, CudaText — MPL-2.0): их бинарники можно спокойно вложить в сборку, приложив копирайт. Notepad++ же под **GPLv3** — это copyleft. Сам по себе GPL нам не помеха: F4 запускает `notepad++.exe` как внешний процесс, никакой линковки с нашим кодом нет, «заражения» проприетарной части не происходит (доктрина mere aggregation). Кусается другое — **распространение бинарника**: вложи мы Notepad++ в архив, и по GPLv3 §6 придётся тащить за собой полный текст лицензии и **исходный код именно той версии** (или письменный оффер его выдать), версия к версии. Плюс «Notepad++» — **товарный знак**: имя и логотип в чужой сборке без спроса — уже история про юристов, а автор (Don Ho) за этим следит пристально.

Возни и рисков ради одного редактора — непропорционально много. Поэтому Notepad++ мы **не раздаём**, а даём кнопку «**Установить Notepad++**» (меню «Audion TC», команда `em_audion_npp_install`, вращпер `Run\Npp-Install.cmd` → `Get-App.ps1 -App npp`): пользователь одним кликом качает свежий портативный Notepad++ **с официального первоисточника** (github `notepad-plus-plus`) прямо в `Apps\Notepad++`. Сборка при этом ничего не распространяет — значит ни обязательств GPL, ни претензий по бренду не возникает. Самый народный редактор Windows оказался самым хлопотным для включения — такая ирония строгой лицензии.

Свои темы Audion Dark. У Textadept — `Ware\textadept-userdata\themes\audion-dark.lua` (флаг `-u` наводит на неё), у Lite XL — `Ware\lite-xl-userdata\colors\audion_dark.lua` (обёртка `Run\LiteXL.cmd` задаёт `LITE_USERDIR`). Обе строят фон/акценты/подсветку из цветов `AudionDark.ini`: фон `#1A2126`, синтаксис в оранжевом/бирюзе/стальном, MD-заголовки — тёплый оранжевый. Lite XL несёт 100+ language-плагинов (cmd/ini/json/yaml/toml и десятки языков) и шрифт JetBrains Mono с кириллицей.

Обновление самого Total Commander

`Scripts\Update-TotalCommander.ps1` берёт официальный установщик (ссылку ищет на странице загрузки, версия в адресе меняется), распаковывает `INSTALL.CAB` и копирует **только те файлы, которые в сборке уже есть**. Ни одного нового файла не появляется, поэтому не возвращаются ни два десятка чужих языков, ни штатные панели, ни справка — всё, что из сборки убрано намеренно.

Отдельным списком `$keepOurs` защищено то, что мы заменили своим: `Wcmicon2.dll` и `Wcmicons.inc` — библиотека значков (у штатной другой набор и другие номера, после перезаписи половина кнопок показала бы чужие картинки), плюс конфиги и ключ.

Проверено на переходе 11.56 → 11.58: обновилось 11 файлов, 21 уже совпадал, 60 файлов дистрибутива справедливо пропущены как не наши.

Каталог и снимки

Карта — это ещё и каталог: записи без программы в `Apps` показываются пультом как «можно поставить». Отсюда пункты «Обновить одну...» и «Поставить программу, которой нет...» — один и тот же список, разделённый по наличию папки. Плагины в эти списки не попадают: у них свой раздел меню. А «Обновить одну...» показывает не всё установленное, а только то, у чего вышла новая версия, — по кешу проверки обновлений (ниже).

`Scripts\Backup-Config.ps1` снимает состояние настроек в `Ware\Backup\<дата>`: `Wincmd.ini`, `Usercmd.ini`, `ShellDetails.ini`, папки `Bars`, `Menus`, `colors` и карту. История и `Apps` не входят. Восстановление сначала снимает текущее состояние (откат обратим) и отказывается работать при живом ТС.

Кэш значков панелей (`*.br2`, `*.br96` — ТС заводит по спутнику на каждый масштаб) в снимок не попадает: он составлял три четверти веса — 878 КБ против 324 КБ, — а восстанавливать его незачем, панель описана в `.bar`, картинки менеджер соберёт заново.

Заметку скрипт спрашивает сам и один раз переспрашивает на пустой ответ: в списке снимков видны только дата и она, а дата через месяц не говорит ничего. Пульт поэтому ничего не спрашивает от себя, а `Backup-Config.cmd` без аргументов делает самое частое — снимает снимок. Ключей знать не нужно. Это штатный ответ на «вышел свежий Total Commander»: распаковали поверх, восстановили снимок.

Программы, которые обновляют себя сами или куплены по лицензии, остаются в карте **без source** — с комментарием, чтобы никто не начал искать им источник (так помечен XviD4PSP).

Проверка обновлений

`Get-App.ps1 -Check` опрашивает источники и запоминает, у кого вышла новая версия, — чтобы пульт не показывал в «Обновить одну программу» то, что и так свежее. Итог кладётся в кеш `Ware\State\update-check.json` (кому какая версия установлена и доступна, устарело ли), а пульт читает его и открывается мгновенно, никуда не ходя. Кеш живёт **сутки**: старше — пульт считает его несвежим и показывает все программы (лучше лишнее в списке, чем спрятать то, у чего на деле вышло обновление), а живая подпись пункта «Проверить обновления» зовёт запустить снова и говорит, когда проверяли и сколько нашли.

Опрос сетевой и идёт программа за программой — на медленном канале это до полутора минут (с токеном вдвое меньше, см. ниже). Чтобы паузу не приняли за зависание, `-Check` сразу пишет ожидаемое время и печатает по строке на каждый источник, а подпись пункта в пульте несёт ту же оценку в скобках («опрос до ~90 сек» / «~минуту» с токеном).

Разведку версии ведут те же функции, что и загрузку, только без скачивания: у каждого источника есть `Resolve-<источник>` (тег и имя файла), а `Get-<источник>File` = разведка плюс загрузка. Так проверка и обновление не расходятся. У GitHub на проверке быстро выбирается лимит API (60 запросов в час), но разведка, как и загрузка, сама переходит на HTML страницы релиза — просто медленнее.

- **установленную версию** берём с самого надёжного, что под рукой: свойства `exe` (FileVersion), затем версия в имени файла, затем в имени папки;
- **доступную** — из ответа источника: у GitHub это тег (надёжнее имени ассета: `yt-dlp.exe` версии не несёт, а тег — да), у winget строка из `winget show`, у остальных — имя файла со страницы;
- **сравниваем покомпонентно**, добивая нулями: иначе `100.0` и `100.0.0` — одна версия — разошлись бы (у `[version]` лишний разряд считается старше), и актуальная программа мигала бы как устаревшая.

Серая зона. У части программ версию не узнать, не скажав: имя файла её не несёт (`SysinternalsSuite.zip`, `PCI-Z.zip`, `7z2501-extra.7z`, `XviD4PSP-win64.zip` — числа без точек). Такие всегда показываются в списке обновлений — по решению владельца: спрятать реальное обновление хуже, чем показать лишнее. XviD4PSP при этом качается по-настоящему: имя zip стабильно, а источник ведёт на свежую версию через главную winnydows (главная → свежая страница релиза → zip), поэтому «Обновить» всегда берёт последний выпуск.

Плагины не проверяются намеренно. Их мало, обновляются редко, и у них свой раздел меню — гонять на них сеть незачем. `-Check` берёт только программы.

Кеш — рантайм: он про конкретную машину и в поставку не входит (как и `Ware\State\downloads`). У конечного пользователя он свой — первый пульт покажет всё и предложит проверить.

Токен GitHub — по желанию, для скорости. Аноним ограничен 60 запросами к GitHub API в час, и на проверке (около двадцати репозиторий) при повторных прогонах лимит кончается — разведка уходит на HTML страницы релиза, вдвое медленнее. Токен поднимает лимит до 5000/час: на живом замере (медленный канал) проверка ускорилась с 69 до 42 секунд. Берётся он по порядку: переменная окружения `GITHUB_TOKEN` / `GH_TOKEN` (для разовых прогонов и CI), затем файл `Ware\State\github-token.dat`. Задаёт его пульт («GitHub-токен для проверок»): ввод скрытый, значение сразу шифруется **DPAPI** под текущего пользователя Windows (`Set-GithubToken.ps1`). Токен нужен только на машине, где часто гоняют проверку; конечным пользователям и так хватает анонимного лимита.

Токен в поставку не входит — это важно. Файл `github-token.dat` живёт в `Ware\State` (как кеш и `downloads`) и при раздаче сборки не копируется. Даже попади он на чужую машину — DPAPI не даст его прочитать: расшифровать может только тот же пользователь Windows и только там, где шифровал. Обе защиты складываются: файла в раздаче нет, а был бы — бесполезен. Всё равно перед тем, как отдать сборку кому-то, стоит убедиться, что `Ware\State\github-token.dat` в ней отсутствует. Пульт умеет это сам — пункт «Подготовить к раздаче» в разделе «Уборка» снимает токен, кеш проверки, тестовый `Test.ini`, временные загрузки и кэш значков панелей (`Bars\*.br*`), всё по путям от корня той сборки, где запущен. Свои файлы `State` (`bar-slots.ini`, `menu-icons.txt`, `new_file_names.ini`) и настройки он не трогает.

Меню «Audion TC» — единый вход

Раньше наши операции жили в меню «?» рядом со справкой Гислера. Теперь у них своё меню в верхней строке — плоское, с разделителями, под именем сборки. Расчёт простой: пультом на fzf будут пользоваться немногие, а меню видно списком, и помнить команды не нужно. Всё управление собралось здесь — от установки программ до тем оформления (темы и F4-редактор — хвост того же меню, см. ниже).

Меню и пульт — один код, а не две реализации. Пункт, которому нужен выбор (поставить одну программу, обновить одну, удалить), не открывает своё окно, а **прыгает в нужную секцию пульта**: команда `em_audion_apps_*` зовёт `Run\Panel.cmd <действие>`, тот — `Launcher.ps1 -Direct <действие>`. Ключ `-Direct` уводит пульт мимо главного меню сразу в дело: где нужен выбор — открывается тот же fzf-список, что и из пульта; где действие однозначно (обновить всё,

собрать версии) — сразу рамка с выводом. Пункты без выбора (создать атлас, задать токен) зовут свой скрипт напрямую. Так меню не дублирует пульт, а даёт в него короткие входы.

Почему через `Panel.cmd`, а не прямой командой: `%COMMANDER_PATH%` в поле параметров TC не разворачивается (см. «Ловушки»), а одна обёртка на все входы избавляет от двух десятков однострочных `.cmd`. Список действий `-Direct` держит `Launcher.ps1`: добавить пункт — значит дописать ветку там и строку в меню (пункты приложений — между маркерами `Sync-Apps`).

Наборы и Base-фундамент

`Audion TC Base` (бывшая Release) — не урезанный продукт, а релизная база: из неё движок разворачивает любой состав. Три набора-списка в `Scripts\builds\`:

- `initial.txt` — исходный (редакторы + TeraCopy + резиденты) = сам Base;
- `audion.txt` — выбор автора (текущий мастер);
- `ultimate.txt` — всё качаемое и хранимое.

Формат — имя секции `apps_map.ini` на строку (`#`/`;` — комментарий), тот же, что у `-InstallList`. Поэтому «установить набор» это `Get-App.ps1 -InstallList Scripts\builds\<набор>.txt` — отдельного кода нет, переиспользован установщик по списку. Пункты меню (`em_audion_apps_setaudion` и пр.) зовут его через `Panel.cmd` → `Launcher.ps1 -Direct`, с плашкой-подтверждением на скачивание и очистку.

`Scripts\Clear-Apps.ps1` сносит `Apps\` в ноль, `Profiles\` не трогает — очистка обратима по составу: поставил набор заново, преднастройка вернулась. «Установить исходный состав» = Clear-Apps плюс набор initial.

`Ultimate` и мастер — не раздача, а эталоны: Ultimate полон как источник значков программ (`menu-icons.txt` → `Wcmicon2.dll`) и как проверка, что всё скачивается. Наружу уходит Base (и Payload — её overlay-близнец). VS Code ставится и обновляется своими кнопками (`Run\VSCode-Install` / `Run\VSCode-Update` → портативный VS Code в `Apps\VSCode`), а не картой загрузки: он крупный и нужен не всем, поэтому в комплекты не входит — как и Notepad++.

Карта версий приложений

`Ware\State\app-versions.json` — каталог доступных версий: плоский `"группа.имя": "версия"` по всему набору из `apps_map.ini`. Одна карта на все сборки — она про то, что

вообще можно поставить и какой свежести, а не про конкретную машину.

Этим она и отличается от кеша проверки (`update-check.json`, см. выше). Кеш — рантайм одной машины: что установлено против доступного, устарело ли; живёт сутки, в поставку не идёт. Карта версий — общий справочник: наполняется пунктом **«Собрать версии приложений»** (`Get-App.ps1 -Catalog` опрашивает источники и пишет доступное) и дописывается при каждой удачной загрузке. Из неё берут версии документов — атлас и `Apps.md`.

`-Catalog` не смотрит в программу и работает **без TC и без Apps** — только по ссылкам из карты (для этого `$root` находится и когда рядом нет `TOTALCMD64.EXE`: нужно в пакете-наложении, где своей копии TC нет). Это отдельная команда именно потому, что задача другая: не «что у меня устарело», а «сними свежие версии всего каталога».

Обновление сверяется с картой, а не с программой. «Обновить все» пропускает то, что в карте значится актуальным, — но только если программа **реально стоит**: папка в `Apps` есть и не пуста. Иначе чистый Release, где карта уже полна, а `Apps` пуст, поставил бы ноль (на этой дыре и попались). Правило: пропуск — лишь когда есть что пропускать.

Версию, которую по сети снять не удалось (nightly, редкий сайт, серая зона), обновление и установка берут **принудительно**: лишняя загрузка дешевле пропущенного нужного. Отсюда и правило для документации — самовольное обновление программы приравнивается к принудительному: в карте останется то, что снял `-Catalog`, а не то, что поставили мимо.

Конструктор установки

`Scripts\Build-Installer.ps1` собирает `install.hta` — окно выбора, которое любую сборку (Release, Рабочую, Ultimate) превращает в нужную конфигурацию. Пункт меню **«Установить выбранные приложения»** его и открывает.

Внутри — программы каталога галочками по группам, готовые наборы (**Админ / Программист / Креатор**, плюс **Всё / Ничего**), и у каждой программы с нашим профилем вторая галочка **«профиль»** (по умолчанию включена: ставим настроенной; снял — поставится чистой). Наборы — простые списки имён секций в `Scripts\sets\*.txt`, правятся текстом. Кнопка «Установить выбранные» сверху пишет отмеченное в `Ware\State\install.list` (строкой `имя` или `имя|noprofile`) и зовёт `Run\Install-Set.cmd` → `Get-App -InstallList`. Актуальное пропустится по карте версий, поэтому «Всё» выбирать не страшно, — а отдельной кнопки «Установить всё разом» намеренно нет: только осознанный выбор.

HTA — это Internet Explorer. `mshta` рендерит движком Trident, и часть современного CSS там мертва: flex `gap` игнорируется молча (отступы — только `margin`), поэтому знак в верхней

плашке отбит от кнопок вертикальной чертой на полях, а не зазором. Проверять правки `install.hta` — запуском самого HTA, не в браузере: Chromium простит то, что IE сломает.

Почему HTA, а не страница с кнопкой скачивания: HTA имеет доступ к файлам и запуску, поэтому кнопка сама пишет список и стартует установку — без промежуточных файлов и без запущенного Total Commander.

Атлас программ и конфигов

`Scripts\Build-Atlas.ps1` собирает из одной карты две вещи: `Docs\Atlas.html` — наглядную раскладку, и `Docs\Apps.md` — ту же таблицу для чтения в редакторе. Задача — чтобы сборка не была чёрным ящиком: видно каждую программу, её источник, куда ставится, в каких сборках числится (●/·), какой версии (из карты версий) и какие наши конфиги на неё заведены — сниппеты разворачиваются на месте. Верхние плашки-счётчики кликаются как фильтры. Пункты меню «Открыть атлас» и «Создать атлас заново» — показать готовый и пересобрать.

Сниппеты конфигов атлас вставляет текстом, поэтому двоичные файлы отсеивает заранее: `Get-Snippet` узнаёт их по BOM UTF-16 или нулевому байту и пропускает — иначе бинарные `.cfg` foobar затащили бы в HTML управляющие символы.

`-Catalog`, `-Check` и атлас работают по одной карте `apps_map.ini`, но не пересекаются: каталог снимает версии, проверка ищет устаревшее на машине, атлас рисует картину. Одни данные — три взгляда.

Пять цветовых тем

В разделе оформления меню «Audion TC» пять пунктов тем: «Тема: Audion» (личная тёмная, `colors\Audion.ini`), «Audion Dark» (дефолт, `AudionDark.ini`), «Audion Sea Blue» (`AudionSeaBlue.ini`), «Audion Emerald» (`AudionEmerald.ini`) и «Audion Light» (светлая, `AudionLight.ini`). Лицо темы задают три цвета, остальное общее:

- **Заголовок** — `DarkHighlight`: заголовки колонок и фон выбранного диска. Везде один, **Sea Blue** `#004360` — фирменный синий Audion: шапка панелей узнаётся сразу, в какой теме ни сиди.
- **Выделение отмеченных** — `MarkColor`, единственный акцент, которым темы и различаются: у Sea Blue — тот же `#004360` (в тон заголовку), у Dark — тёплый контраст

синему `#56240B` (RGB 86,36,11), у Emerald — изумруд `#034905`. Тёмная тема по умолчанию — **Audion Dark**.

- **Курсор** — `CursorColor`, подсветка строки под фокусом: у тёмных ярко-голубой `#00B5FF` (читается на тёмном фоне), у светлой — Sea Blue `#004360`.

Тема в TC — две палитры в одном файле: `[Colors]` (светлая) и `[ColorsDark]` (тёмная), какую показать — решает `DarkMode` (0 светлая, 2 тёмная). Переключение — две вещи: скопировать эталон в `colors\Current.ini` и выставить `DarkMode`.

`Scripts\Set-Theme.ps1 -Audion|-Dark|-SeaBlue|-Emerald|-Light` (враппер `Run\Set-Theme.cmd`, команды `em_audion_theme_*` вне маркеров `Sync-Apps`): кладёт палитру в `Current.ini`, закрывает TC, правит `DarkMode` (после закрытия — живой TC затёр бы конфиг при выходе), запускает заново и переставляет bullet-маркер в меню на выбранную. При живых программах из `Apps` — отказ: перезапуск осиротил бы их, а идущее копирование/перенос и вовсе оборвал.

Светлую собрали из палитры дизайн-гайдов: кремовый фон, тёмный текст, акценты по типам файлов. Значения цвета в TC — десятичный BGR (`R + G*256 + B*65536`), поэтому темы правятся скриптом, а не на глаз. Файлы тем в cp1251 (имена фильтров русские) — Edit-инструментом не трогать (см. выше).

Оформление в меню плоское и без иконок — только bullet-маркер у активной темы и активного редактора: пять тем, редакторы по наличию — от одного Notepad до шести (см. выше) и два служебных пункта — «Очистить ключи» (`Set-GithubToken.cmd -Remove`) и «Восстановить настройки по-умолчанию» (`Reset-Defaults.ps1` с перезапуском TC).

Кодировки

Что	Кодировка
<code>.bar</code>	UTF-16 LE с BOM
<code>.ini</code> , <code>.MNU</code> , <code>.LNG</code>	cp1251
<code>.ps1</code>	UTF-8 с BOM
<code>.cmd</code>	UTF-8, текст только латиницей
<code>.lua</code> (темы Textadept/Lite XL)	UTF-8

Инструменты

`Run\Launcher.cmd` — пульт на fzf. Кнопка на него стоит в главной панели. Пункты разложены по группам — «Программы», «Сборка», «Настройки: сохранить и вернуть», «Уборка»; заголовок группы выбирается как обычная строка, но пульт на нём просто перерисовывает меню (проверка по первым символам, до вызова).

Вид. Название сборки с путём — не отдельная строка над списком, а метка на самой рамке (`--border-label`), рамка скруглённая, счётчик найденного стоит справа в строке ввода. Список занимает окно целиком (`--height=100%`), а экран перед отрисовкой чистится: иначе меню уезжало вниз вслед за выводом прошлой команды, сверху оставалась пустота, и всё выглядело прижатым к левому нижнему углу. Пункты пронумерованы сквозной нумерацией с отступом; заголовок группы — линейка во всю ширину с названием по центру, тёмно-оранжевым, тем же цветом, что и метка на рамке.

Цвет живёт только в отображаемой строке. fzf с `--ansi` раскрашивает список, но на выход отдаёт выбранное уже без escape-кодов, поэтому у каждой строки меню хранится пара `Line` / `Plain`, и действие ищется по чистому тексту. Сам цвет — ANSI-256 (166): в палитре консоли тёмно-оранжевого нет, `DarkYellow` там олива. На откате с PowerShell 7 на 5.1 VT может быть выключен, и коды напечатались бы мусором — там берётся ближайший штатный цвет.

Дочерние экраны. Вывод команды получает такую же шапку: название дела по центру, под ним «← Назад — Esc» и линейка. Возврат — по Esc или Enter (`[Console]::ReadKey`, с откатом на `Read-Host`, если ввод перенаправлен): прежний `Read-Host` ждал именно Enter, хотя рука тянется к Esc — им же закрывается и сам пульт. Вложенные списки подписаны тем же названием на рамке, а первым пунктом в них стоит «← Назад». Название берётся из подписи пункта до тире и без хвостовой скобки: «Применить настройки (ТС должен быть закрыт)» становится делом «Применить настройки».

Пункты «показать» — каталог программ, версии плагинов, что лежит в `Apps`, что сохранено — ничего не запускают, поэтому их вывод не льётся в консоль, а открывается в такой же рамке, что и списки выбора: то же название на рамке, прокрутка, поиск по строкам, выход по Esc (`Show-View`). Вывод дочернего скрипта для этого ловится в переменную — `ps.cmd` отдаёт его через `stdout`, и кириллица не страдает: обе стороны говорят в UTF-8. Дела, которые качают и ставят, остаются как были — их вывод должен идти вживую, иначе за длинной загрузкой не видно, что происходит.

Сам `fzf.exe` лежит в `Scripts\bin` и обновляется тем же пультом: в карте он записан группой `tool` — панели её не собирают, кнопки не появляются, а загрузчик работает с записью как со всеми. В архиве он одним файлом, поэтому папка не подменяется целиком и `.bak` не остаётся.

Функция-заголовок называется `Divider`, а не `Group`, и это не вкусовщина: `group` — штатный алиас `Group-Object`, а алиасы в PowerShell разрешаются раньше функций. Заголовок молча становился `$null`, и меню падало на «A null key is not allowed in a hash literal».

Каждое действие со снимками имеет свой wrapпер, чтобы ключи не требовалось знать: `Backup-Config.cmd` — сохранить, `Restore-Config.cmd` — показать список и вернуть выбранное по номеру, `Backup-List.cmd` — просто посмотреть. Те же три пункта есть в пульте.

Сброс к эталону — отдельно от снимков. Снимок хранит *твоё* состояние (датированная копия в `Ware\Backup`), а эталон — наши золотые версии, вшитые в саму сборку: `Wincmd.backup` (конфиг с нашими настройками и размером Lister), `colors\Audion.ini` (цвета) и `Usercmd.backup` (меню). Пункт пульта «Сбросить к эталону» возвращает конфиг, цвета и меню к ним, а перед откатом сам снимает снимок текущего — сброс обратим. Отказывается работать при живом ТС (он перезаписал бы конфиг при выходе) и молча пропускает эталон, которого нет. В мастере и Release сброс полный. Пакет-наложение теперь тоже накатывает эталоны целиком (`Wincmd.ini`, `Usercmd.ini`, `History.ini`) — `overlay.ini` упразднён, расчёт на чистый Total Commander: почти все ставят на свежескачанный, а не поверх своего. Так исключён рассинхрон, где копия настроек в `overlay.ini` разъезжалась с настоящим `Wincmd.ini`.

Правило про эти файлы: `Wincmd.ini`, `Usercmd.ini`, `.bar`, `.MNU` — в cp1251/UTF-16, и **их нельзя трогать текстовым Edit-инструментом**: он переписывает файл в UTF-8 и превращает кириллицу (заголовки колонок, заметки) в мусор. Править только через PowerShell с явной `GetEncoding(1251)` — прочитать байты, заменить латинские ключи, записать теми же байтами.

`Run\ps.cmd` — PowerShell 7, откат на 5.1. `psc.cmd` — команда в окне, которое не закрывается. `psadmin.cmd` — то же с правами администратора. `Apply-Config.ps1` — то, что нельзя делать при живом ТС (лестница DPI, справка, горячие клавиши). `Update-Zapret.ps1` — сверяет версию и отдаёт работу загрузчику, а сам снимает winws, службу и драйвер по ключу `-StopOnly`, когда загрузчик об этом попросит. Если драйвер залип в памяти, замена не начинается: прежняя версия остаётся на месте, а в окне написано, что нужна перезагрузка.

Тот же обновлятор есть и отдельно от сборки — `E:\TOOLS\Zapret Updater`, портативный набор для тех, у кого нет Audion TC: сам находит папку рядом, а не найдёт — ставит Zapret с нуля.