Building Asset Packs: End-User Guide

This guide explains how to produce asset packs for one specific game version, verify them, and prepare a complete `/app0` set. The primary workflow uses the `ampr_pack_gui.py` desktop application. Equivalent PowerShell commands are provided at the end for recovery and diagnostics.

> **The main tracing limitation**
>
> Traces **do not contain all game data**. They show only APR reads that actually happened during the recorded scenarios. They omit unvisited levels, other modes and languages, some DLC and rare branches, as well as access through paths that were not intercepted, such as `mmap`.
>
> An automatically generated TOML profile is only a starting point. Before packing, adapt it to the real game: add the remaining required files and directories, or intentionally keep them as ordinary loose files. Even a long trace does not prove that an unobserved file will never be required.

1. What you need

- Windows and Python 3.11 or newer with Tcl/Tk. Tkinter is included in a normal Python installation from python.org.
- A complete, unchanged local copy of the game's `/app0` directory.
- `ampr_emu.index` built for that exact `/app0` copy.
- One or more traces. Each run directory must contain `ampr_commands.bin` and its matching `ampr_emu.index` next to each other.
- Enough free space for both the original files and the generated packs.

Install the packer's dependency from PowerShell in the repository root:

```
python -m pip install -r tools/requirements-pack.txt
```

Do not combine a trace from one game version with an index from another. `fileId` depends on AMPRIDX3 records, so retain the exact index used to record each trace beside that trace.

If the complete local copy does not have an index yet, build it once and do not change it until all traces and packs are complete:

```
python tools/build_ampr_index.py "profiling/title/app0"
```

Recommended working-directory layout:

```
profiling/title/
├── app0/                          complete source game
│   └── ampr_emu.index
├── traces/
│   ├── startup/
│   │   ├── ampr_commands.bin
│   │   └── ampr_emu.index
│   ├── level-and-combat/
│   │   ├── ampr_commands.bin
│   │   └── ampr_emu.index
│   └── travel/
│       ├── ampr_commands.bin
│       └── ampr_emu.index
├── profiles/
└── packed/
```

2. Recording useful traces

Trace recording uses a ready-made **debug emulator build** with APR command recording enabled.

Record several separate scenarios:

1. Cold start through the main menu and loading a save.
2. Loading several different levels or large regions.
3. Fast travel, cutscenes, combat, and intensive streaming.
4. Revisiting a region.
5. Additional modes, languages, and installed DLC that the finished set must support.

After each normal exit, copy `ampr_commands.bin`, `ampr_emu.index`, and, for diagnostics, `ampr_emu.log` into a separate run directory. A journal interrupted by a crash may be incomplete.

More runs improve geometry and cache recommendations, but do not replace the manual game-content review.

3. Starting the GUI

On Windows, double-click:

```
tools\ampr_pack_gui.cmd
```

Or launch it from PowerShell:

```
python tools/ampr_pack_gui.py
```

The GUI selects Russian for a Russian system locale and English for every other locale. Use the language selector at the top of the window to change it without restarting. To force a language at launch, use:

```
python tools/ampr_pack_gui.py --lang en
python tools/ampr_pack_gui.py --lang ru
```

Fill in the five paths at the top of the window:

- **Trace directory** — a common directory below which the GUI finds all `ampr_commands.bin + ampr_emu.index` pairs.
- **Complete /app0 copy** — original game files, not a directory from which source files have already been removed.
- **ampr_emu.index file** — the index for this complete game copy.
- **TOML profile** — where generated rules are saved, for example `profiling/title/profiles/title.toml`.
- **Pack output directory** — a separate result directory, for example `profiling/title/packed`.

Click **“1. Generate profile from traces”**. The GUI merges all discovered runs and saves these files next to the TOML:

- `title.report.md` — a human-readable report;
- `title.metrics.json` — detailed input metrics;
- `*.include.txt` lists when needed.

The JSON describes input access and recommendations. It is not a performance measurement of the completed packed runtime.

4. Adapting the profile to the real game is mandatory

After generation, click **“Load from profile”**. **“2. All paths selected for packing”** then shows every explicit `include` and `include_from` from traced `compress` / `store` rules together with previously saved manual additions. This shows the generated selection instead of an empty manual-only list.

Compare this list with the actual `/app0` contents and required game scenarios, then edit the selection:

- **“Add files…”** selects one or more individual files;
- **“Add directory…”** adds the entire directory as `path/**` ;
- **“Add pattern…”** adds a relative path or glob manually, such as `dlc/**` or `localization/en/**` .
- **“Remove selected”** removes a path from the effective selection. For a trace-derived path, its original rule and parameters remain intact and the GUI writes a final loose override;
- **“Load from profile”** rebuilds the complete editable list from the selected TOML and its safe relative `include_from` files.

Selected paths must be inside the configured `/app0` . The GUI writes them to `title.manual.include.txt` and adds a managed TOML rule. This does not override hard exclusions for `eboot.bin` , PRX/SPRX files, indexes, and system directories.

Parameters for new manual paths

GUI parameter	Meaning	When to use it
Mode: `compress`	Compresses independent blocks with LZ4 HC level 12 by default; an incompressible block is stored RAW	Initial choice for most game data; slower to build than `fast` , but usually smaller
Mode: `store`	Avoids LZ4 CPU work but places data in a seekable pack with CRC	Already-compressed video, audio, and archives after testing

GUI parameter	Meaning	When to use it
Access: `mixed`	Balances bounded random access and merging of adjacent reads	Recommended initial choice
Access: `random`	Optimizes small, non-sequential reads	Tables, scripts, configuration, and frequently used small files
Access: `streaming`	Places sequential blocks densely	Large video/audio or data proven to be sequential
Block	Independently readable and decodable block size from 16 KiB to 1 MiB	Start with `64KiB`; use larger blocks for streaming and smaller blocks for fine random reads

Mode, **Access**, and **Block** apply only to new paths that were not part of the loaded trace-derived rules. Existing rule geometry is preserved rather than flattened; edit the corresponding `[[rule]]` in TOML to change those parameters.

The GUI adds `force_pack = true` to the new-path rule. Explicitly selected files will not automatically return to loose because of weak compression; their incompressible blocks remain RAW. Hard system exclusions still take priority.

For already-compressed streaming video or audio, a reasonable starting point is `store`, `streaming`, and `256KiB`, but change geometry only with a subsequent in-game A/B test. If different groups need different parameters, save the main list through the GUI, then add separate `[[rule]]` entries manually. Rules are applied top to bottom; the last matching rule wins.

Pay particular attention to:

- levels, maps, and biomes absent from recorded runs;
- every supported language, voice-over, and subtitle set;
- DLC, bonus modes, new-game paths, and final chapters;
- data used by cutscenes, menus, credits, and error recovery.

When the review is complete, click **“2. Save path changes”** and select the coverage confirmation checkbox. The checkbox does not prove completeness; it prevents accidental packing immediately after trace-profile generation.

5. Building and verifying packs

Click **“3. Pack and verify”**. The GUI performs, in order:

1. Build pack volumes and `ampr_assets.index`.
2. Verify every block, CRC, and decode operation.
3. Compare every reconstructed packed file byte-for-byte with source `/app0`.
4. Print the manifest summary.

While packing, the GUI log shows `ampr_pack.py` progress lines with the current phase, overall percentage, processed file count, processed source bytes, current path, and elapsed time. Progress also updates within one large file, so an HC12 compression pass does not appear stuck.

The operation succeeds only if all four stages complete without an error. Use **“Verify existing packs”** and **“Show summary”** to repeat those operations separately. Full output remains in the bottom section of the window.

Do not remove source files after one successful `verify`. Verification proves pack integrity, not full scenario coverage or interception of every read method.

6. Removing originals that are stored in packs

After making a backup, the GUI can remove from the selected `/app0` only files that the final manifest marks PACK. Click **“Remove PACK files from /app0…”**.

The GUI shows the file count and total size before removal. After confirmation, it runs complete pack verification again and compares the packed content with every original byte-for-byte. Removal does not begin if the manifest, a pack, the source size, or source content does not match.

Removal parameters:

Parameter	Purpose
Complete /app0 copy	Directory from which originals are removed; paths come only from PACK records in the selected manifest
Pack output directory	Contains the verified `ampr_assets.index` and volumes; LOOSE records are never removed

Parameter	Purpose
Coverage review confirmation	Required: offline verification does not prove that traces covered the whole game or that a file is not read through `mmap`
Also remove directories that become empty	Removes only directories that are actually empty after file removal; the `/app0` root is never removed

The operation also refuses to remove `eboot.bin`, PRX/SPRX/SELF/ELF files, AMPR indexes, system directories, symlinks/reparse points, or any path outside the selected `/app0`, even if an incorrect custom TOML marked it PACK.

> Removal is permanent. Afterwards, `verify --root` can no longer compare packs with their source files. Keep a separate backup. Packed files can also be restored with `unpack` while the correct index and every volume remain intact.

Before launching the game, make sure `ampr_assets.index`, its `.runtime` file, and every volume are already included in the final `/app0` set. Source removal does not copy those files there.

7. Files to deploy with the game

Deploy one matching set to `/app0`:

- the source `ampr_emu.index`;
- `ampr_assets.index`;
- `ampr_assets.index.runtime`;
- every `.pak` named by the manifest;
- every file left loose.

Do not mix an index, sidecar, and volumes from different builds. Runtime settings are bound to the manifest build ID. Replace the set only while the game is not running.

For initial tests, retain all originals and use a debug emulator build with packed-file support. Removing an original is acceptable only after testing all required game paths and only for a PACK file that is not accessed through an unintercepted method. Never remove modules, executables, indexes, or settings.

8. Runtime configuration parameters

The TOML profile contains a `[runtime]` section. Packing converts it into an `ampr_assets.index.runtime` file bound to that manifest's build ID:

```toml
[runtime]
decoded_cache_bytes = "128MiB"
physical_cache_bytes = "32MiB"
workers = 4
latency_reserve_workers = 1
```

Parameter	Purpose and valid values
`decoded_cache_bytes`	Cache for decoded blocks. Must be a multiple of 16 KiB; `0` disables it. A larger cache helps repeated reads but consumes the shared internal pool
`physical_cache_bytes`	Cache for physical pack pages. Must be a multiple of 16 KiB; `0` disables it. It helps when several blocks reuse the same page
`workers`	Packed-runtime worker threads: `1..16`. This is not `[pack].workers`, which controls PC-side compression
`latency_reserve_workers`	Workers reserved for latency-sensitive reads: from `0` through `workers - 1`

The GUI uses these values when it builds packs. To change runtime settings only, edit `[runtime]` in the TOML and update the sidecar without recompressing:

```
python tools/ampr_pack.py runtime-config --index "profiling/title/packed/ampr_assets.index" --c
```

Deploy the updated `ampr_assets.index.runtime` and restart the game. A damaged sidecar or one from another build ID blocks manifest loading. A missing sidecar means compiled defaults are used.

9. Console testing

At minimum, test startup, a new game, several saves, level transitions, fast travel, cutscenes, streaming, language changes, DLC, revisits, load cancellation,

and normal game exit.

For diagnostic measurements, use a ready-made debug packed emulator build with detailed binary tracing and verbose logs disabled.

Save each identical scenario's `ampr_emu.log` separately. Do not sum periodic snapshots: they are cumulative. p50/p95/p99 values are histogram upper bounds, and worker time is elapsed work time, not CPU time.

```
python tools/analyze_ampr_log.py "profiling/title/runs/baseline/ampr_emu.log" --tail 0
```

Finally compare the completed set in the supplied release packed build against the loose baseline for load time, frame time, memory, and correctness. The debug build has its own overhead and is not suitable for final performance evaluation.

10. Equivalent commands without the GUI

This example uses two traces and runs from the repository root:

```
$caseRoot = Join-Path (Get-Location) 'profiling/title'

python tools/ampr_pack_profile.py generate `
  --trace "$caseRoot/traces/startup/ampr_commands.bin" "$caseRoot/traces/startup/ampr_emu.index'
  --trace "$caseRoot/traces/travel/ampr_commands.bin" "$caseRoot/traces/travel/ampr_emu.index" `
  --output "$caseRoot/profiles/title.toml" `
  --report "$caseRoot/profiles/title.report.md" `
  --metrics "$caseRoot/profiles/title.metrics.json" `
  --pattern-mode exact --cache-sim --cache-max-touches 250000 --overwrite
```

After generation, manually add rules for all remaining required files and directories. For example:

```
[[rule]]
action = "compress"
include = ["dlc/**", "localization/en/**", "levels/not_visited/**"]
block_size = "64KiB"
layout = "mixed"
force_pack = true
```

Keep safety exclusions last because the last matching rule wins. Then build and always verify with `--root`:

```
python tools/ampr_pack.py pack --root "$caseRoot/app0" --ampr-index "$caseRoot/app0/ampr_emu.inc
python tools/ampr_pack.py verify --index "$caseRoot/packed/ampr_assets.index" --root "$caseRoot/
python tools/ampr_pack.py inspect --index "$caseRoot/packed/ampr_assets.index"
python tools/ampr_pack.py list --index "$caseRoot/packed/ampr_assets.index" --json
```

The `pack` command writes progress to `stderr` and keeps the final JSON alone on `stdout`, preserving scripts that parse that JSON. Add `--no-progress` to suppress progress output.

`list` shows what remains loose. `--self-contained` prevents explicitly selected incompressible files from automatically returning to loose and stores their blocks RAW, but it **does not select every game file automatically**.

Preview the removal plan first. Without `--confirm`, nothing is changed:

```
python tools/ampr_pack.py remove-packed-sources --index "$caseRoot/packed/ampr_assets.index" --r
```

After backing up and reviewing the list, perform removal. `--confirm` authorizes the irreversible change; `--remove-empty-dirs` additionally removes only directories that became empty:

```
python tools/ampr_pack.py remove-packed-sources --index "$caseRoot/packed/ampr_assets.index" --r
```

Changing block sizes, layout, or file selection requires repacking and another `verify`. Changing only `[runtime]` does not require recompression:

```
python tools/ampr_pack.py runtime-config --index "$caseRoot/packed/ampr_assets.index" --config '
python tools/ampr_pack.py inspect --index "$caseRoot/packed/ampr_assets.index"
```

Retain the final TOML, `*.include.txt`, `.runtime`, report, and test results as the profile for this exact game version. After a game update or resource-layout change, record new traces, repeat the manual coverage review, and rebuild packs.