

Static State Auditor Pro

©Copyright Digital Software/Johan 'Zarkow' Munkestam

Version: 1.0.0, includes Static State Auditor 1.0.1 (Unity 6.3 LTS, 6000.3 and newer; editor-only, any render pipeline)

Introduction

Static State Auditor finds every static field, auto-property and event that survives "Enter Play Mode" without a domain reload. Pro fixes them: it generates the reset code, shows you after each play session which statics actually changed and which still hold a value, keeps a list of the ones you mean to leave alone, and can write a reset method straight into your own class. Everything happens in the same window; Pro adds one row of controls and two columns.

This package contains the free package in full. If the free package is already in your project, importing Pro updates it to 1.0.1 in place.

Getting started

Open Window > Digital Software > Static State Auditor. The Pro row sits under the classification toggles. The first time you press Generate, Ignore or Settings, a settings asset is created at Assets\StaticStateAuditorPro.asset; it holds the options described below and the ignore list, and it is yours to commit.

Demo: open Assets\DigitalSoftware\StaticStateAuditorPro\DemoScene\ProDemoScene, enable Edit > Project Settings > Editor > Enter Play Mode Settings with "Reload Domain" unticked, and press Play twice. The on-screen score climbs from 125 to 150 although the field is declared as 100, the registry list grows, and the run counter from the free demo climbs. Stop, and the Last play column shows what changed and what survived. Press Generate reset code, play again: everything starts over.

The workflow

1. Scan. Read the mutable and event rows; they are the candidates.
2. Play once, stop. The Last play column now says, per member, whether the value changed during the session and whether it still holds a value afterwards. "Survived" rows are the ones that would leak into the next run. Tick "diff rows only" to see just those.

3. Decide per row. Intentional: select it, type a reason, press Ignore selected. Needs a reset: either leave it for the generated file or press Insert reset method to put the reset into the class itself.
4. Press Generate reset code. Rescan after the recompile; covered rows show "generated" in the Reset column.
5. Play again and read the diff. When nothing you care about survives, turn domain reload off for good and keep the generated file in version control. Regenerate after you add, rename or remove statics; the file tells you in the console when a type it knows has gone.

The Pro row

Generate reset code

What: writes one C# file (Assets\StaticStateResets.g.cs by default) with a method marked `RuntimeInitializeOnLoadMethod(SubsystemRegistration)`. Unity runs it before the first scene of every play session, with or without domain reload. The method puts every mutable static field, static auto-property and static event from your own runtime assemblies back to its declared value: the literal from the declaration where there is one (`= 100`, `= 9.81f`, `= "none"`), otherwise the type default.

When: after every scan that changed the list, and after refactors. It is safe to press as often as you like; the file is overwritten each time and never touches your own files.

Why: this is the fix Unity recommends, done for the whole project in one click, without you writing or maintaining it.

How it finds members: by name, through reflection, from inside a file wrapped in `#if UNITY_EDITOR`. That has three consequences. Private members and events are reachable without changing their visibility. Renaming or deleting a type never breaks compilation; the generated method logs one warning naming the stale entry so you regenerate. Players carry none of this code; a built game never ran without a domain reload in the first place.

What it leaves out: package assemblies and editor code (not yours to reset), ignored rows, types that already have their own `SubsystemRegistration` reset (they handle themselves; see the option below), `[ThreadStatic]` members, members of generic types (reset those in the type itself), and compiler-generated statics. The Console lists what was skipped and why.

Insert reset method

What: inserts a SubsystemRegistration method into the class of the selected row, in that class's own source file, with one assignment per static member of the class: fields back to their declared literal or default, events to null, a selected readonly collection to Clear(). The method is placed right after the opening brace of the class; the file's encoding and line endings are kept.

When: for classes you want to own the reset of, typically a singleton or a manager whose reset should sit next to its fields and be reviewed in code review; also for anything the generator leaves out, such as a generic type. For everything else the generated file is less work.

Why: some teams want no generated file in the repository, or want the reset visible where the state is declared. After the recompile, rescan: the class shows "yes" in the Reset column and the generator skips it from then on.

A get-only auto-property cannot be assigned from a method; the inserted code leaves a comment on that line so you can add a private setter. A readonly reference without a Clear method also becomes a comment.

Ignore selected, reason

What: marks the selected member as intentional. It disappears from the list and the counts, the generator leaves it out, and the reason is stored in the settings asset with the date. Select an ignored row (with "show ignored" ticked) and the same button reads Unignore selected.

When: a cache that is rebuilt lazily on first use, a lookup table filled once at declaration, a debug flag, a static that every Awake assigns anyway. The rule of thumb: ignore it when the second play session is correct even though the value carried over.

Why: the mutable count is only useful when it means "still to decide". Writing the reason down keeps the decision from being revisited by the next person.

show ignored

Shows the ignored rows again, greyed, with their reason in the Ignored column. Use it to review or undo decisions.

diff rows only

Shows only members that changed during the last play session or still held a value when it ended. This is the shortest route from "7 mutable" to "these 3 matter". Needs one play session first.

Settings

Selects the settings asset in the Inspector. The options are described in their own chapter below.

The summary label

Shows when the last play session ended and how many members changed and survived, out of how many were watched. "No play session recorded yet" until you have played once since the editor started.

The columns

Reset: as in the free package, plus "generated" when the generated file covers the member and the class has no reset of its own.

Ignored: the reason given when the row was ignored, or "(no reason)". Empty for rows that are not ignored.

Last play: empty before the first play session. Afterwards one of "unchanged" (same value at start and end, and the end value is the type default), "changed" (the value differed at the end but ended up at default again), "survived" (the end value is not the default: it leaks), or "changed, survived". When the class has its own reset or the generated file covers the member, the label reads "survived (reset)" in green: the value is there at exit, and the reset clears it before the next session starts. Hover the cell to see the value at play start and at play end. Reference collections report their item count, Unity objects their name, or "destroyed" when the object no longer exists, events their handler count.

The play-mode diff

When play mode starts, Pro reads the value of every mutable, event and reference member in your assemblies at the "before scene load" point: after all SubsystemRegistration methods have run, before any Awake. That is why a singleton with a working reset shows null at play start and a destroyed object at play end. When play mode stops it reads them again and compares. The summary label says which start point was used; "after first

frame" means the early hook did not fire and the start values already include the first Awake and Start calls. The result survives domain reloads within the editor session, so the diff works whether "Reload Domain" is on or off; with it on, every row is "unchanged" by definition, which makes it a quick way to confirm that a reset file does its job before you turn the reload off.

Two caveats. Reading a static value runs the static constructor of a type that has not been initialised yet, the same as any access to the type would. And a mutable field holding a plain object is reported as "survived" whenever it is not null; the diff cannot see inside an arbitrary object, so judge those rows by the tooltip and your knowledge of the class.

The settings asset

Assets\StaticStateAuditorPro.asset. Created on first use; the package never overwrites it.

Generated file path

What: where Generate writes the reset file. Default Assets\StaticStateResets.g.cs, which compiles into Assembly-CSharp.

When to change it: when your project has no Assembly-CSharp, or when you want the file inside a specific assembly definition folder. Keep it in a runtime assembly folder, never under an Editor folder: the method must run when play mode starts, and the #if UNITY_EDITOR inside the file already keeps it out of builds.

One method per type

What: on, the entry method calls one private method per class, each named after the class. Off, every reset is inline in the entry method with a comment per class.

When: on (the default) for readability and smaller diffs in version control. Off if you prefer a flat file you can read top to bottom.

Clear readonly collections

What: also emits Clear() for "reference" rows whose type has a parameterless Clear method: static readonly List, Dictionary, HashSet, Queue and the like.

When: on when your readonly collections are filled during play, for example registries that components add themselves to in Awake. Off, the default, because a readonly collection that is filled at declaration (a lookup table) would be emptied too, and nothing would fill it again. Check the Last play column: a readonly collection that "survived" with a growing

item count is the case this option is for; a collection whose count is the same at start and end is a table, leave the option off or ignore that row.

Include types with own reset

What: also generates resets for members whose class already has a `SubsystemRegistration` method.

When: on only if you want the generated file to be the single list of everything that is reset, for review purposes. Off, the default, because the class is already handling itself and a second reset of the same member is redundant.

Log resets

What: the generated method logs one line per member it resets.

When: on for one play session to see the resets happen in the Console; then off, it is noise. Takes effect after the next Generate.

Play-mode diff enabled

What: takes the two snapshots described above.

When: on while you are cleaning up. Off when you are done, or in a very large project where the extra scan at play start and stop (tens of milliseconds per thousand statics) is noticeable. The last result stays visible until the editor is restarted.

Diff includes packages

What: also watches statics in package assemblies.

When: on only when you suspect a package of carrying state between sessions and want evidence to report; the row count grows into the thousands. Off by default.

Ignored

The ignore list itself: type name, member name, reason, date. Edit reasons here, delete entries to unignore in bulk, or paste entries from another project. Entries are matched by type and member name, so a renamed type leaves a stale entry; remove it.

Limitations

Members marked `[ThreadStatic]` are listed but never reset. Static members of generic types cannot be reached by name; insert the reset method in the type instead. Events with custom add and remove accessors have no backing field to clear. Members in precompiled

DLLs are listed but have no source to insert into. Initial values that are not plain literals (new expressions, method calls, other members) are reset to the type default; adjust the generated line or move the value into a reset method of your own.

Further support

Head over to <https://community.digitalsoftware.se/> or the Discord at <https://discord.gg/fyqtxZt>