Page contents

- Prepare a reproducible load order
- Isolate one conflict
- Choose the least invasive resolution
- Test the patch and diagnose failures
- Scope and sources

Offline copy of Records, load order and compatibility patches .

Fallout 4: inspect records and resolve plugin conflicts

Documentation status: In progress. **Source checked:** 2026-11-11 against the xEdit project manual. **Technical validation:** the procedure below has not been run against a local Fallout 4 installation for this publication. The xEdit manual uses some Fallout 3 screenshots; its load-order and record-inspection concepts apply to xEdit's Fallout 4 mode, but do not treat the pictured forms as Fallout 4 examples.

This chapter is for a mod author whose plugin edits a Fallout 4 record also edited by another plugin. The goal is to identify the *specific fields* in conflict, decide which values should survive, and test a small compatibility patch. A colored conflict marker is evidence of an override relationship, not proof of a bug. The xEdit conflict manual explicitly distinguishes intentional overrides from harmful conflicts.

Prepare a reproducible load order

Make a separate test profile containing the base game, required DLC, your plugin, and one suspected conflicting plugin. Record the enabled plugin names and order before opening xEdit. Back up your plugin and test save. Start xEdit in its Fallout 4 mode (`FO4Edit` or the `-fo4` mode selector), select the plugins you need to inspect, and wait until loading finishes. The xEdit overview explains that Fallout 4's load order comes from files maintained by the mod manager; xEdit displays that order but does not

change it. If you reorder plugins, close and reopen xEdit so the inspection reflects the new order.

The left tree groups records by plugin and type. Select a record to compare its versions in the right pane. The rightmost column is the last loaded override among the selected plugins. Read the individual fields, not just the record's color. The values that matter depend on the mod: an item may have a changed name, value, model path, keywords, or gameplay properties. A patch that copies one desired field but unintentionally restores another mod's old value can create a new incompatibility.

Isolate one conflict

Use **Apply Filter to show Conflicts** to narrow the view, then choose one record affected by both plugins. The xEdit overview lists this command in the navigation tree's context menu. Write a small comparison before editing:

Question	Record in your notes
What is the original form?	Plugin and form identifier shown by xEdit
Which plugins override it?	Every relevant column, in load order
Which fields differ?	Field names and each plugin's intended value
Which result should the player see?	The expected behavior in game
Can load order alone produce it?	Yes, no, or still unknown

For example, if one plugin changes an item's display name and another changes its value, neither plugin's whole-record override necessarily preserves both intents. Compare the fields in xEdit and test the outcome in game before deciding on a patch. If the later plugin already carries both intended changes, no extra patch may be needed. If a conflicting change is intentional and desirable, leave it alone.

Worked comparison pattern

The values below are a **schematic example**, not a captured Fallout 4 record or a tested mod. It shows the decision you should make from your own xEdit columns:

Column	Display name	Value	Interpretation
Original master	Standard name	10	Baseline for comparison
Earlier plugin A	Renamed item	10	A intends a name change
Later plugin B	Standard name	25	B intends a value change but carries the old name
Proposed patch	Renamed item	25	Combines the two stated intents, if no other fields matter

Do not copy only these two cells without inspecting the whole record. B may also change a keyword, model path, component list, or script property that its author intended to preserve. Conversely, A's name may deliberately depend on a different asset or condition. Write down each changed field and the reason for keeping it. The patch should be the last relevant override, with A and B as dependencies when their forms are referenced. Test the three unpatched cases before judging the patched result.

Choose the least invasive resolution

The xEdit manual presents four broad choices: accept the conflict, change load order, make a patch plugin, or stop using one incompatible mod. Use a load-order change only when one plugin's existing override already contains the full desired outcome. A load-order change cannot combine two distinct values by itself. Use a patch when the final record must carry selected fields from both plugins.

Create a **new** patch file for your own compatibility work rather than editing someone else's plugin. In xEdit, right-click the conflicted record and choose **Copy as override into...**, then choose a new file. The manual patch procedure describes this flow and the resulting master dependency. Give the patch a name that identifies the two mods it reconciles. Let xEdit add required masters, and verify the patch's master list before publishing. Load the patch after both source plugins so its override is the final one.

In the patch's record column, bring over only the fields needed for the intended combined behavior. Inspect the entire resulting record after each change. If the copied record references a form defined by another plugin, that plugin must remain a

dependency. A patch that loads without its required master is not a valid release. Avoid changing form identifiers or master flags as a casual conflict fix; the xEdit manual warns that FormID renumbering can break saves and dependent plugins.

Automatic merged-patch output is a starting point, not a final answer. The xEdit patch discussion says the automated method cannot resolve every record type or infer the author's preferred result. Review every generated override that affects your mod before distributing it.

Test the patch and diagnose failures

Save the patch, restart xEdit with the intended load order, and confirm that your patch is the final column for the target record. Recheck the field values and read the Messages tab for load-time problems. The **Checking for Errors** command does not replace that review: the xEdit overview describes it as a check against record definitions, while missing-reference information appears during loading.

Then test in Fallout 4 with a fresh save and an existing test save. Compare three cases: your mod alone, the other mod alone, and both with the patch. Use a reachable item or location so the expected effect is observable. If the combined result fails but the record values look correct, investigate scripts, asset paths, initialization timing, and other records. A record-level patch cannot fix every interaction.

Package the patch with a short dependency list, the tested game build, the two source-plugin versions, and a description of the exact conflict it resolves. State that changing or removing either source mod requires renewed review. Keep the test profile and notes so a later game or mod update can be checked against the same scenario.

Scope and sources

This chapter documents the record-inspection and compatibility-patch workflow. It does not claim that every Fallout 4 conflict is a simple last-override case, and it does not provide a tested patch for a named pair of mods. The next validation step is a concrete two-mod example with captured xEdit record values and in-game results.

- xEdit overview and Fallout 4 load-order behavior
- xEdit conflict detection and manual patch procedure

- xEdit release and game-mode information