Guide contents

What you will make

Before opening the editor

Know the three names you will see

1. Start a new plugin

2. Duplicate a vanilla weapon

3. Add a forge recipe

4. Reopen and enable the plugin

5. Test the weapon, then test normal acquisition

6. Package and retest without development leftovers

Troubleshoot the earliest failed step

Where to go after this project

Sources and limits

Documentation: In progress **Quick start: In progress**

Offline copy from ModDocs. Source links remain clickable.

Skyrim modding: make your first usable weapon

This guide walks through a small **Skyrim Special Edition / Anniversary Edition PC** mod: a new sword based on an existing one, plus a forge recipe that lets a player make it. You will create a plugin, enable it, test both the item and its recipe, and package it for another installation. You do **not** need to draw, write code, or install SKSE for this project.

> **Scope:** This is a source-reviewed Creation Kit workflow, not a claim that we ran these steps in your exact game build. Keep the Creation Kit matched to your Skyrim edition and test the result in game. Original Skyrim (2011), Special Edition/Anniversary Edition, VR, and different storefront releases have different tool/runtime requirements. The steps below target the PC Special Edition family.

What you will make

File or record	Purpose	Example used here
Plugin	Stores your new records and their links to Skyrim's data	`ModDocsPracticeSword.esp`
Weapon record (`WEAP`)	Defines the sword's name, damage, model, value, and other behavior	Editor ID `MD_PracticeSword`; in-game name **ModDocs Practice Sword**
Constructible Object (`COBJ`)	Defines a recipe and where it appears	Editor ID `MD_RecipePracticeSword`; output `MD_PracticeSword`

The example IDs and plugin filename are names **you create**, not Bethesda records you must find. The sword will reuse a vanilla model, so this first version needs no custom mesh or texture. A weapon record alone does not automatically appear in a shop, loot table, or forge. The recipe is what makes this particular example obtainable through normal play.

Before opening the editor

1. Confirm that you have Skyrim Special Edition or Anniversary Edition on PC. Install and launch it once. Record the game build you are testing, especially if you later install script-extender mods.
2. Install the matching Creation Kit from Steam. Launch it once and allow it to locate the game data. You need the base game's `Skyrim.esm`; this project does not need DLC records.
3. Back up saves you care about. Create a separate mod-manager profile or use a disposable test save. Keep a copy of your `.esp` outside the game folder before each substantial edit.
4. If you already have many plugins, disable unrelated mods in the test profile. A clean profile makes a missing item or a conflicting override easier to diagnose.

The game does not need to be running while you edit. Creation Kit writes the plugin; Skyrim reads it on the next launch. Creation Kit File menu.

Know the three names you will see

Editor ID is the editor's unique, human-readable identifier, such as `MD_PracticeSword`. **Name** is the text the player sees, such as “ModDocs Practice Sword.” **FormID** is the hexadecimal number the game assigns to a loaded record. Its leading load-order digits can change when plugins move. The game console's `player.additem` command takes a **FormID**, not the Editor ID or the `.esp` filename.

A **base record** defines what a weapon is. A **placed reference** is one instance of an object in a cell. Duplicating a weapon base record creates a distinct item without changing every vanilla sword. Editing the original record instead creates an override and can affect all uses of that sword.

---

1. Start a new plugin

1. Open Creation Kit and choose **File** → **Data**. Select `Skyrim.esm` so the editor loads the base-game records. Do not set `Skyrim.esm` as the active plugin. Let the editor finish loading before you edit anything.
2. For a brand-new mod there is no existing `.esp` to mark active yet. Make the new weapon record in the next section, then choose **File** → **Save** and enter `ModDocsPracticeSword.esp` when prompted. The File menu reference documents this first-save behavior.
3. After that first save, check that the file exists in the game's `Data` folder. Copy it to a separate working-backup folder. If the save created a differently named file, use that actual filename consistently from here on.

When continuing later: in **File** → **Data**, select `ModDocsPracticeSword.esp`, choose **Set as Active File**, and load its required master(s) with it. If you load your plugin without making it active, the next save may create another plugin instead of updating this one. Do not make a copy of `Skyrim.esm` or edit that master directly.

2. Duplicate a vanilla weapon

1. In the **Object Window**, open **Items** → **Weapon** (some Creation Kit views have deeper subcategories). Filter for a familiar, ordinary **Iron Sword** weapon record. Select the weapon itself, not a recipe, static display model, or placed world reference.
2. Right-click the weapon and choose **Duplicate**. This is important: opening the original and changing its damage would override a base-game item. The Creation Kit's weapon walkthrough demonstrates duplication, although its art steps were written for an older Skyrim edition.
3. Open the duplicate. Set its **ID / Editor ID** to `MD_PracticeSword` and its player-facing **Name** to `ModDocs Practice Sword`. If the editor asks whether an ID change should create a new form, verify the result afterward: there must be one new `MD_PracticeSword` record and the original must retain its old ID. The exact prompt can vary by editor version.
4. Note the copied **Damage** value, then raise it by **2** for an obvious test. For example, if the copy says `D`, enter `D + 2`; do not rely on a fixed vanilla value from this page. You may also change **Value** to make the inventory comparison clearer.
5. Leave the copied model, first-person model, sounds, animation, and keywords alone for this first pass. Reusing the original art removes asset packaging from the test. Accept the record, choose **File** → **Save**, and name the new plugin `ModDocsPracticeSword.esp` if prompted.
6. Filter the Object Window for `MD_PracticeSword`. Open it and verify the new Name and Damage. Then inspect the original Iron Sword: its value should still be unchanged. If the original changed, undo or discard that edit and duplicate again.

At this point the new `WEAP` record exists, but the player has no normal way to find it yet. You can smoke-test it with the console, but adding a recipe makes the mod useful without a console command.

3. Add a forge recipe

1. In the **Object Window**, open **Items** → **Constructible Object**. Find a *vanilla Iron Sword forge recipe* as a starting point. Look at the record's output and

workbench before you duplicate it; recipes for tempering or smelting are different operations.

2. **Duplicate** that forge recipe. Give the duplicate a unique Editor ID, `MD_RecipePracticeSword`.
3. Set **Created Object** (the output) to your `MD_PracticeSword` weapon and **Created Object Count** to `1`.
4. Keep the forge **Workbench Keyword** from the comparable sword recipe. Inspect the **Required Item** list. For a first test, retain the same ordinary base-game ingredients as the copied forge recipe; write them down so you can carry them into the test save.
5. Inspect every **Condition** on the copied recipe. A skill, perk, or other condition can hide it from a new character even when the plugin works. Either keep the conditions and test with a character that satisfies them, or remove an unwanted copied condition in **your duplicated recipe**. Never change the vanilla recipe to make this test easier.
6. Confirm that the output still points to `MD_PracticeSword`, not the original Iron Sword. Save the plugin. Reopen the new recipe and verify the output, workbench, ingredients, and conditions survived the save.

A Constructible Object record is what makes an item craftable; the weapon's Name or model cannot do that alone. Creation Kit crafting-category reference. If your Creation Kit build labels fields differently, use the copied vanilla recipe as the field-by-field reference rather than guessing a keyword or condition.

---

4. Reopen and enable the plugin

1. Close Creation Kit. Reopen it, choose **File** → **Data**, mark `ModDocsPracticeSword.esp` **Active**, and load it. Check that both `MD_PracticeSword` and `MD_RecipePracticeSword` are present. This catches a save to the wrong plugin before you start the game.
2. Close Creation Kit. In your test mod-manager profile or the game's plugin selection mechanism, enable `ModDocsPracticeSword.esp`. Check that it is listed as enabled and that required masters are available. A file merely existing in `Data` is not proof that the game loaded it.
3. Start Skyrim and load a disposable save. Do not use a long-term save as the sole test. Keep unrelated mods disabled until the sword and recipe work.

5. Test the weapon, then test normal acquisition

Direct record check. Open the game console and type:

```text
help "ModDocs Practice Sword"
```

Look for a result identified as a **WEAP** record. Copy its displayed hexadecimal **FormID**. The number is specific to your loaded game and plugin order, so there is deliberately no fixed number in this guide. Then enter:

```text
player.additem <FormID from the WEAP result> 1
```

Replace the angle-bracket text with the actual hexadecimal value. Check the inventory for **ModDocs Practice Sword**, equip it, and compare its displayed damage with a normal Iron Sword on the same character. The console result also proves the plugin's weapon record loaded. The Creation Kit weapon tutorial uses the same `help` then `player.additem` check.

Recipe check. Give the test character the recipe's required materials legitimately or through your normal test setup. Visit a forge, open its crafting menu, and look for **ModDocs Practice Sword**. If you kept skill or perk conditions, use a character that meets them. Craft one, then check the item in inventory. This test proves more than `player.additem`: the `COBJ` output, workbench, ingredient list, and conditions must agree.

Save your observations in a short log: plugin filename; game edition/build; whether `help` found a `WEAP`; whether `player.additem` worked; whether the forge showed the recipe; and whether crafting produced the new sword. A test is complete only when you checked the outcome in game, not just in the Creation Kit.

6. Package and retest without development leftovers

For this example, the installable mod is just `ModDocsPracticeSword.esp`, because it reuses base-game models and contains no scripts or new assets. Copy that file into a small release folder; do **not** include `Skyrim.esm` or extracted game assets. Include a

readme listing the supported Skyrim edition, what the plugin adds, any masters, how to enable it, and how to remove it.

Create a **second clean test profile**, install only the release copy of the plugin, enable it, and repeat both tests. This matters because a mod can appear to work on the development installation by accidentally reading loose files or another enabled plugin. If you later add a custom mesh, texture, script, sound, or localization file, include every needed file in its expected path and repeat this clean-profile test.

Troubleshoot the earliest failed step

Symptom	Check first	Next action
No `.esp` appeared after saving	File → Data active-file state and the actual filename	Save to a new plugin; verify the file in `Data`, then make it active when reopening.
The original Iron Sword changed	Whether you edited the source record instead of a duplicate	Remove or revert the accidental override; create a separate duplicated `WEAP`.
`help` finds no new `WEAP`	Plugin enablement, edition, and the in-game Name you searched	Reopen the `.esp` as active, verify the record, then re-enable the plugin and restart Skyrim.
`player.additem` fails	Whether you copied the **FormID** from the `WEAP` result	Use that loaded-game FormID; the Editor ID is not a console item number.
The sword exists but not at a forge	`COBJ` output, forge workbench, ingredients, and conditions	Reopen your duplicated recipe and compare it to the vanilla forge recipe; test with a character meeting its conditions.
Missing model or wrong appearance	Model fields or omitted external files	Restore the copied vanilla model for this project; custom art needs a separate packaging test.

Symptom	Check first	Next action
Works with your mod list, fails in a clean profile	Missing master or accidentally relied-on assets/plugins	Inspect dependencies and include your own files; do not redistribute base-game assets.

For a conflict that appears only with other plugins enabled, inspect records and load order in xEdit. It can show which plugin wins an override; do not blindly “clean” or change another author's plugin because a tool highlights a record.

Where to go after this project

World placement: use the Creation Kit's Cell View and Render window to place a reference to your new item in a test cell. Verify it can be reached and picked up. This is separate from defining the weapon base record.

Custom art: model and texture files are external assets referenced by plugin records. The older retexture tutorial explains the relationship, but check paths and export settings against your game edition. Do not copy extracted Bethesda assets into a public mod package without rights to distribute them.

Papyrus: add scripts only when the base record and recipe already work. Keep both source and compiled files, attach the script to the intended form/reference, and test its event in game. Papyrus logging helps diagnose script behavior; it is not a crash log.

SKSE / native DLLs: a normal item mod does not require SKSE. Native plugins are a different, runtime-sensitive route; start from the Skyrim.dev first-plugin tutorial only when records and Papyrus cannot implement the behavior you need. Match the exact Skyrim runtime and SKSE version.

Sources and limits

This walkthrough combines the Creation Kit Steam listing, File menu documentation, weapon duplication/test tutorial, and crafting-category documentation. The weapon tutorial and parts of the wiki predate the current Special Edition tools, so field labels and prompts should be verified in your installed Creation Kit. The example plugin and records have **not** been built and tested locally by ModDocs. This guide describes one

basic plugin project; quests, NPCs, custom art, Papyrus, cross-edition conversion, and SKSE plugins each need their own complete workflows.