Guide contents

Choose module or add-in

Make one visible change

Export and test the installed result

Sources / References

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

Offline copy from ModDocs. Source links remain clickable.

Dragon Age: Origins modding

Dragon Age: Origins has a dedicated **Dragon Age Toolset** for making modules, add-ins, areas, items, conversations and scripts. The community-maintained Toolset Wiki documents the editor and its game-specific terms. The workflow below is for **Origins**; it does not imply that Dragon Age II or Inquisition use the same tools.

> **Quick start:** install the Origins Toolset for a matching PC game installation, create a uniquely named add-in module, change one resource, export the module, enable it in-game, and verify the change in a disposable save.

Choose module or add-in

The module-creation reference distinguishes a **standalone module**, which can be its own campaign, from an **add-in** that extends the original single-player campaign. In **File → Manage Modules**, create a new module with a unique UID. The UID becomes part of file and folder names, so choose it once and keep it consistent. Record the assigned StringID range; the toolset wiki warns that it matters when exporting to another database.

For a gameplay tweak to the main campaign, set the **Single Player** module as parent in the Hierarchy settings. A standalone campaign leaves the parent unset. This choice controls what resources the module can see and how it appears in the game. Avoid creating a new module merely to experiment with an existing campaign resource without deciding which relationship you need.

The wiki also warns against using `module_core` as the event script for some add-ins extending existing modules. Follow the module properties and current examples for the exact script setup instead of copying a standalone campaign configuration into an add-in.

Make one visible change

Start with a resource you can easily find in a test save, such as a custom item. Give new resources names beginning with your UID, as the module guide recommends. This avoids collisions with other add-ins. Inspect an existing resource for the fields and references it needs; do not overwrite the original record if a new resource can prove the workflow.

For a map or quest, first build one small area, a start location and the minimum conversation or trigger needed to enter it. The Toolset tutorial index links module, area and scripting walkthroughs. Test the area in a fresh run before adding several branches of dialogue. Resource ownership matters: the wiki explains that a core resource can be visible across campaigns, whereas an add-in normally inherits its parent. A script owned by the wrong module may be absent when that resource is loaded.

Export and test the installed result

Export or package the **module**, rather than assuming the toolset's working database is what another player will receive. Check the Toolset Wiki's export and packaging instructions for your toolset version. Install the exported copy into a clean game profile, enable it in the game's **Downloadable Content** menu if it is an add-in, and verify the resource using a disposable save. If it works only in the editor, inspect resource ownership, exported dependencies and the module's parent setting.

Keep toolset database backups; the module guide notes that a module cannot simply be deleted from within the editor. Include only original or redistributable assets. Document the base game, expansion and toolset versions used for testing.

This guide is **unfinished**: it describes the module decision and first test loop, but a complete, current-version exported item or playable-area example still needs step-by-step resource fields, scripting and a clean-install verification.

Sources / References

External project and documentation links cited in this guide. Check the upstream release and version before following version-sensitive steps.

- Toolset Wiki
- module-creation reference
- module properties
- Toolset tutorial index