Page contents

- Separate the pieces of a quest
- Quest lifecycle and script methods
- Alias and objective failure checks
- Dialogue and release boundary

Offline copy of Quest stages, objectives and aliases .

Fallout 4 quests: stages, objectives and aliases

Source status: This chapter maps the Fallout 4 `Quest` and `ReferenceAlias` interfaces in the maintained Papyrus Index and the preserved Creation Kit quest-tutorial hierarchy, checked 2026-11-11. The tutorial mirror currently rejects direct page fetches from this research environment, so this chapter does not invent its editor clicks or claim a completed quest build. The API entries below are source-reviewed; the workflow remains untested in game.

Separate the pieces of a quest

A quest record is more than a title and description. It owns a lifecycle, stages, objectives, aliases and optional dialogue or script fragments. A **stage** is a progression marker; an **objective** is a displayed task the player can see; an **alias** binds a quest role to an actor or reference. A stage can run script fragments, but setting a stage does not by itself guarantee that every objective is shown or completed in the way you intend. Plan the progression as a small state table before editing:

State	Trigger to inspect	Objective expected	Reference or actor needed	Exit condition
Not started	Start rule or explicit script call	None	Any startup aliases	Quest initialized

State	Trigger to inspect	Objective expected	Reference or actor needed	Exit condition
Initial task	Stage transition	First objective visible	Target alias resolves	Player reaches target
Task resolved	Interaction or dialogue	First objective completed	Same target remains valid	Completion stage runs

The table is a design aid, not a prescribed set of stage numbers. Use unique IDs and actual editor data from your project. Do not attach a long dialogue chain until a minimal start, state change and completion can be observed reliably.

Quest lifecycle and script methods

The indexed `Quest` script exposes `OnQuestInit()` after startup aliases have filled and while the startup stage runs. `OnQuestShutdown()` occurs when the quest shuts down and aliases are cleared. `OnStageSet(int auiStageID, int auiItemID)` reports a stage-setting event. These are different observation points: a handler requiring a filled alias belongs after successful startup, while a handler running after shutdown must not assume the alias still resolves. The Quest API index also lists Story Manager events, but which event starts your authored quest requires a separate editor design and version-specific test.

Interface	Parameters and return	Use and limit
`Start()`	`bool`; no arguments	Starts the quest; latent and waits for startup and its fragments. Check the returned result before assuming the quest began.
`SetStage(int aiStage)`	Stage index; `bool` result	Attempts a stage transition. A nonexistent or unsuccessful stage leaves state unchanged. It can wait for startup and stage fragments.
`SetObjectiveDisplayed(int aiObjective, bool`	Objective index and	Shows or hides an objective. `abForce` requests display even if it was

Interface	Parameters and return	Use and limit
`abDisplayed=true, bool abForce=false)`	display flags; no result	displayed earlier.
`SetObjectiveCompleted(int aiObjective, bool abCompleted=true)`	Objective index and completion flag; no result	Marks an objective completed or incomplete. Check the visible journal state after the call.
`GetAlias(int aiAliasID)`	Alias ID; returns `Alias`	Retrieves an alias on that quest. The returned alias still needs to contain the expected runtime reference.

These signatures describe the indexed Fallout 4 API, not a generic Skyrim-only recipe. Keep the source game and script version visible when using a cross-game tutorial.

Alias and objective failure checks

When a quest starts but its interaction never fires, check whether the relevant alias filled with the intended actor or object. The ReferenceAlias index lists events such as `OnActivate(ObjectReference akActionRef)` and `OnDeath(Actor akKiller)` that can be received through an alias, but the event only helps if the runtime alias points to the correct reference and the script is attached in the right place. A missing actor or reference is a different problem from a stage that never changes.

When the journal objective is wrong, inspect stage state and objective state separately. Record which stage was requested, whether `SetStage` returned success, which objective index was displayed or completed, and whether a prior save had already advanced. If multiple stage fragments exist, the SetStage documentation warns that fragments can run concurrently rather than one after another; do not rely on an unstated ordering between them. Use logged evidence and a fresh test save to isolate initialization and repeat behavior.

Dialogue and release boundary

The first dialogue scene chapter covers a speaker alias, greeting, player choices, scene phases and exit paths. The following worked design connects **one** player response to a quest stage and objective. It is an original design derived from the indexed Fallout 4 `Quest` API, not a plugin we built or ran. A Creation Kit fragment must still be compiled, its properties bound, and its exact location checked in the installed editor before this can be called an executable example. The 2016 author tutorial on quest variables through dialogue is a related route for storing a conversation decision; it does not establish that this particular design has been tested.

Work one response through the quest

Use a small quest with a single named NPC alias and the greeting/choice scene from the dialogue chapter. The stage and objective numbers below are **example IDs** for a new plugin. They are not IDs to paste into an existing quest. Assign the final IDs in the editor and keep the table with the project source.

Moment	Example stage	Example objective	Intended player-visible result
Quest starts	10	10 displayed	Player is asked to speak to the NPC
Player accepts the task in dialogue	20	10 completed; 20 displayed	Journal changes to the new task
Player declines	Stay at 10	10 remains displayed, or is intentionally hidden	Conversation ends without accidental acceptance
Task completion trigger	30	20 completed	Quest can finish after a separate completion check

1. **Create the quest's first state.** Give the quest a unique editor ID and its NPC reference alias. Decide how it starts in gameplay; a developer console command is only a diagnostic. Add stages 10, 20 and 30 and objective IDs 10 and 20. On stage 10, arrange to display objective 10. Keep the objective text specific to the target alias so that a wrong or unfilled alias is visible in testing. Confirm the alias can fill *before* adding dialogue conditions that depend on it.
2. **Attach the conversation to the same quest.** Reuse the dialogue chapter's greeting and player choice structure. Restrict the speaker to the intended quest

alias, then make the accepting response point to the scene's exit or next phase. Give the declining response its own deliberate exit. A spoken line or a scene ending does not itself set a quest stage; add an explicit stage transition at the accepting response's authored fragment or other verified trigger.

3. **Request the stage change once.** In a fragment that has an actual, editor-bound `Quest` property, call `SetStage(20)` on that property and check its `bool` result. The Fallout 4 SetStage API says a nonexistent or unsuccessful stage returns `false`. The call is latent and waits for startup and stage fragments. Do not infer success from the dialogue closing. Keep the declining path free of this call unless decline deliberately has its own stage.
4. **Move the journal state.** The stage-20 fragment can call `SetObjectiveCompleted(10)` and `SetObjectiveDisplayed(20)` on the same quest; these are separate objective API operations. An alternative is to place those calls on the accepting response after a successful stage change. Choose one owner for these changes so repeat dialogue cannot create two competing paths. Record that choice beside the quest diagram. Stage fragments on multiple items can start concurrently, so do not depend on one item's fragment finishing before another.
5. **Add a real completion trigger.** Stage 30 should be set by the actual task condition: an activation, item hand-in, area entry or other event appropriate to the mod. Verify the event's alias, reference and script attachment independently. Complete objective 20 only when that condition is met. If the quest should stop or be marked complete, author and test that separately; a stage number alone is not proof of a finished quest.

Do not copy a fragment name, autogenerated property name or VM binding from a tutorial into another plugin. Those are project-specific. The meaningful invariant is that the script invokes a `Quest` instance representing this quest, and that the editor has filled the property before a response can execute. The quest-variable tutorial demonstrates a different dialogue-linked state technique; use a variable when you need to remember a choice independent of a stage, and document its initialization and save behavior.

Verify each boundary on a disposable save

Check	Evidence to capture	If it fails
Start rule and alias	Quest starts; target alias resolves to the expected NPC	Inspect start conditions, alias fill rules and reference persistence before changing dialogue

Check	Evidence to capture	If it fails
Greeting and choice	Only the target NPC says the greeting; both choices appear	Recheck the forced alias and dialogue conditions
Decline	Scene exits; stage 20 is not done	Check for a shared fragment or stage call on the wrong response
Accept	`SetStage(20)` succeeds; objective 10 completes and 20 displays	Check the bound quest property, stage ID, fragment compile result and objective IDs
Repeat talk	Stage and objectives stay in the intended state	Add appropriate conditions or guards; do not assume the scene is one-shot
Completion	Task trigger sets stage 30; objective 20 completes	Inspect the actual trigger event, alias/reference and save state

The Fallout 4 Quest API distinguishes `GetStageDone(stage)` from `GetStage()`: the latter reports the highest completed stage and can mislead when stages are revisited or set out of order. For the acceptance check, inspect the specific stage and the journal objective separately. Retest with a fresh disposable save and a controlled save made before the conversation. Keep a copy of both saves and the exact plugin/build used. No ModDocs in-game test, compiled fragment, voice export, or current editor screenshot has been recorded for this design.

For a quest release, list plugin and master dependencies, scripts and voice or other assets, a reproducible player-facing start condition, stage/objective test path, supported game build, and known save limitations. A quest that progresses only through a developer test command is not yet a validated player-facing quest.