PCA 3D ADVENTURE STUDIO

Action Lists and Events

A practical tutorial and complete command reference

Build responses to clicks, conversations, room changes, triggers and interface controls. Then look up the exact action you need.



Figure 1. The Signal Room, running in the editor. The lessons use a small puzzle so each instruction has an obvious job.

Editor reference: 0.86.3.4
Desktop instructions apply across Windows, macOS and Linux.

How to use this book

Start with the worked lessons if action lists are new to you. They build one small puzzle: use a lever, pick up a key, open a door and talk to an NPC. You can follow the same steps in your own project; the example names are just labels to keep the instructions easy to follow.

The second half is a reference for all 216 action entries found across the shared action catalog and Cutscene Timeline in this build. Each entry explains its job, fields, availability and a practical use. The event reference covers the 18 script subscription names, local events, GUI behaviors and graph flow nodes.

Looking for...	Go to...
A first working interaction	The lever lessons, followed by Test Scene.
A one-time reward or a locked door	The variable, guard and inventory-check lessons.
A button that runs several commands	GUI events and Additional Event Actions.
An event that listens across the game	Script subscriptions and On Event graphs.
A particular command	The alphabetical command index or PDF search.
A problem that seems to repeat itself	Troubleshooting, event ordering and the version notes.

Use the PDF contents and bookmarks to jump between lessons. Command headings include the stored command ID, so searching for either a visible name such as Give Inventory or an ID such as GiveInventory will find its entry.

> The pictures are actual editor and running-preview captures. Some panels are widened or scrolled to keep the relevant controls readable. Names in your own project will differ; the control labels and the order of work are what matter.

Contents

How to use this book . . . 2
1. An event starts the work . . . 5
2. Find the right place to attach it . . . 6
3. Know which editor you are using . . . 7
4. Set up the small puzzle . . . 8
5. Give the lever a Look response . . . 9
6. Make the lever change a fact . . . 10
7. Add visible and audible feedback . . . 11
8. Test the whole response . . . 12
9. Give the key to the player . . . 13
10. Keep a one-time reward one-time . . . 14
11. Attach the greeting to a trigger . . . 15
12. Open a door only with the key . . . 16
13. Entry and exit belong to the room . . . 17
14. Understand room-event order . . . 18
15. Direct clicks and verbs are different routes . . . 19
16. Give a GUI button a working response . . . 20
17. Understand Additional Event Actions . . . 21
18. Preview appearance; test behavior . . . 22
19. Widget input is not a new global event . . . 23
20. Start a conversation from Talk . . . 24
21. Schedule overlapping work on a timeline . . . 25
22. Call the same actions from a script . . . 26
23. Subscribe a script to a shared event . . . 27
24. Emit and receive the custom event . . . 28
25. Use context instead of guessing . . . 29
26. Make this mean the caller . . . 30
27. Pass values, not look-alike text . . . 31
28. Publish a function with editable parameters . . . 32
29. Saving should preserve a finished state . . . 34
30. Wait for the right thing . . . 35
31. Filter completion events carefully . . . 36
32. Connect other designers to the same logic . . . 37
33. Restore controls after a special mode . . . 38

34. Watch an attach-point contact . . . 39
35. Build one dependable puzzle operation . . . 40
36. Diagnose a response that does nothing . . . 41
37. Test the awkward cases too . . . 42
Shared script events: the full reference . . . 43
GUI primary behaviors . . . 49
Graph entries, conditions and flow nodes . . . 51
How to read the action reference . . . 61
Variables, rewards and reusable logic . . . 62
Inventory and two-object interactions . . . 66
Speech, messages, voice and dialogue . . . 70
Characters, movement and animation . . . 82
Objects, descriptions and scene life . . . 93
Cameras, transitions and screen effects . . . 101
Room changes, 2D controls and environmental state . . . 110
Continuous object effects and attachments . . . 120
GUIs, element state and data-driven widgets . . . 132
Control profiles, verbs and pointer behavior . . . 143
Audio, video, FMV and recorded gameplay . . . 150
Quests and objectives . . . 159
Saving, language and project settings . . . 165
Cutscene control and timeline-specific actions . . . 169
Local world events: quick reference . . . 173
GUI mouse events: quick reference . . . 174
Save widget and progress callback data . . . 175
Advanced: structured action data . . . 176
Advanced: a nested conditional in raw data . . . 177
Version notes for these lessons . . . 178
Terms you will see in the editor . . . 179
Alphabetical command index . . . 180
Reference basis and lesson checks . . . 187

1. An event starts the work

An event tells the game that something happened. The player clicked a button. A character entered a trigger. A room finished loading. An action is what you tell the game to do in response: show a message, give an item, move an NPC or change rooms.

An action list is several of those instructions in order. For a pickup, the instructions might be: give the key, remember that it was taken, add five points, and hide the key on the table. The event supplies the starting point; the list supplies the work.

Part	Question it answers	Example
Event	When should this happen?	The door's Use verb is selected.
Action	What should happen?	Show a message.
Action fields	To whom, or with what value?	Text: I need the key. Duration: 2.2 seconds.
Condition	Which path should run?	Does the player have the key?
Context	What happened in this particular call?	The clicked entity ID and selected verb.

A list is not automatically a decision tree. If you put Remove Inventory before Change Scene, the scene change still runs even when there was no item to remove. Put the inventory check in a visual or text script, then call that script from the event.

> Think of the event as the doorbell and the action list as what you do after it rings. Adding a second doorbell does not make the first one ring only once. One-time behavior needs a remembered condition.

2. Find the right place to attach it

The same kind of gameplay command appears in several designers, but the surrounding event is different. Choose the surface that matches the thing the player is doing.

Where	Use it for	How to open it
Object / NPC Verb Actions	Look, Use, Talk, Take and your custom verbs.	Select the entity in Scene Designer; open Verb Actions.
Object / Trigger / Region Events	Entry, exit, direct pointer and hover responses.	Select the scene entity; open its events tab.
Room Events	First entry, every entry, first exit and every exit.	Select the scene record; open Room Events.
Inventory Verb / Inventory Actions	Examine an item or use it with something else.	Open the inventory definition in Inventory Designer.
GUI element Events	Hover/click behavior and extra ordered actions.	Select an element in GUI Designer; open Events.
Dialogue action node	Run a command between dialogue nodes.	Add an Action node in Dialogue Designer.
Cutscene Timeline event	Schedule camera, audio, speech and gameplay beats.	Add/select an event on the timeline.
Script event subscription	Listen for a shared game event.	Open a text script and select its Event Functions.
Visual On Event node	Listen through a visual graph.	Add On Event and choose its event name.

Do not put the same reward in several of these places just to be safe. A door can have both a direct click response and a Use response, and a shared listener can hear the interaction too. Give one of them responsibility for changing the puzzle state.

3. Know which editor you are using

There are two list layouts to recognize. Native room/object inspectors show numbered sections and an Add Next Action button. Choose No Action to remove an instruction. The shared sequence editor, used by GUI Additional Event Actions, gives each row up/down arrows and a remove button.

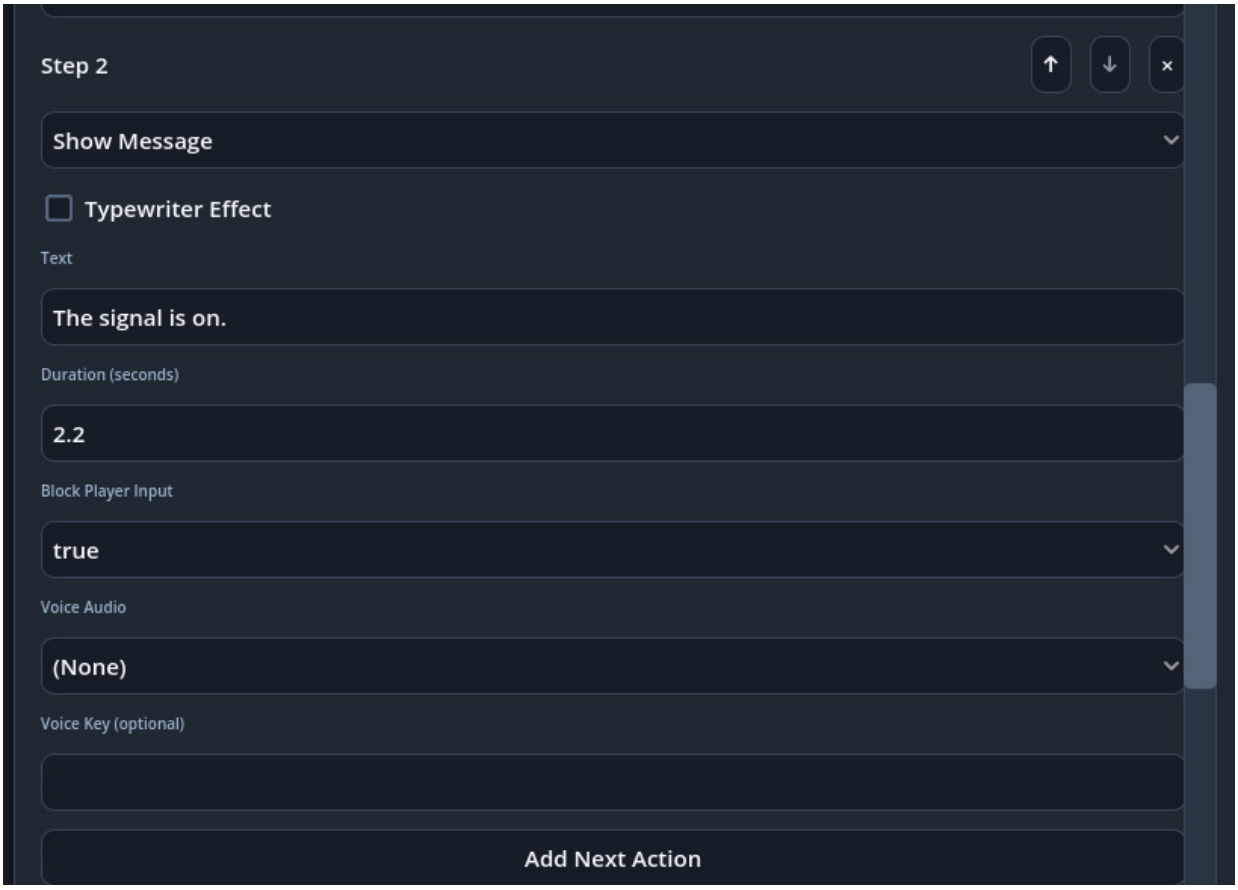

Figure 2. A shared sequence row: action picker, fields, move arrows, remove button and Add Next Action.

In the native inspector, finish your edits with Apply Changes. In the GUI Designer, element-property edits are applied live to the GUI record; still save the project. Graph editors also update their records as you edit, while text scripts have their own Apply Changes control.

Changing an action type changes the fields it needs. Check the target and value again after changing the type. A room picker followed by an entity picker is a dependent selection: choose the room first, then choose the entity from that room.

> Not every picker offers every command. The reference marks Native inspector, Shared list / GUI, Visual graph, Text action wrapper and Timeline separately. When a command is missing from a native inspector, Run Visual Script can call the graph or text script that contains it.

4. Set up the small puzzle

Use a small, working 3D room with a floor, reachable navigation, a player and a default camera. You do not need polished art. A tall brass-colored box can stand in for a lever; a small gold box can stand in for the key. The examples use the bundled character and interface assets where useful.

Record	Example ID	Job
The Signal Room	lab	The starting room.
Brass Lever	lever	Turns on the signal.
Signal Lamp	lamp	A prop whose description changes.
Archive Key	key_pickup	The visible pickup in the room.
Archive Key inventory definition	key	The item the player carries.
Archive Door	archive_door	Checks the bag and changes rooms.
Welcome Area	welcome_trigger	Runs an entry response.
Elias	elias	An NPC with a Talk response.
The Archive	archive	The destination scene.
Archive arrival spawn	from_signal	Where the player arrives.

Create game variables powered, key_taken and greeted as Booleans starting false. Create score and visits as numbers starting 0. Add objective as text if you want a changing objective label.

Use names for people and IDs for stable references. The editor may display Brass Lever(lever); the stored target is lever. Renaming the display name is different from changing an ID used by scripts.

> Keep interaction approach disabled for your very first message test, or use a reachable approach point. That removes walking and collision from the first test. Add approach behavior once you know the event itself works.

5. Give the lever a Look response

1. Open Scene Designer and select Brass Lever.
2. Open the inspector's Verb Actions tab. Scroll to Look Verb - Step 1.
3. Choose Show Message. Enter: A brass lever. Its cable runs to the blue lamp.
4. Set Duration to 2.2 seconds. Leave Block Player Input off for this short observation.
5. Click Apply Changes, save, and use Test Scene. Choose Look, then click the lever.

You should see the observation. If nothing appears, check that the hotspot is enabled and that you are using a mode in which Look is actually selected. In Context mode the object's Default Verb can decide the response instead.

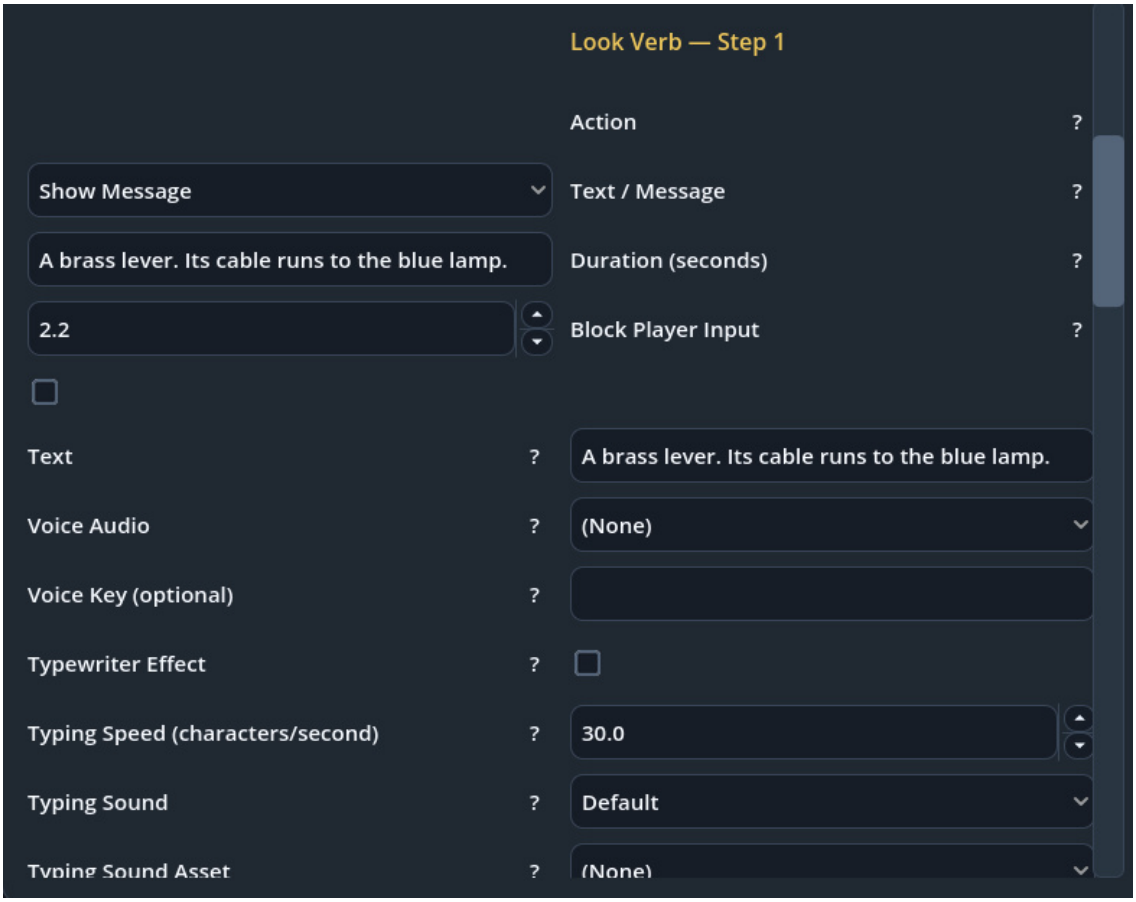

Figure 3. Look uses a Show Message action with the observation and its duration.

> Blank Walk or Talk lists are not missing instructions; they simply have no custom response yet.

6. Make the lever change a fact

Now give Use a persistent result. A message tells the player something happened; a variable lets the rest of the game know it happened.

1. Return to the lever's Verb Actions and find Use Verb - Step 1.
2. Choose Change Variable. Select powered as Variable.
3. Choose Set as Operation. Enter Boolean true as Value.
4. Apply Changes and save. In Test Scene, choose Use and click the lever.

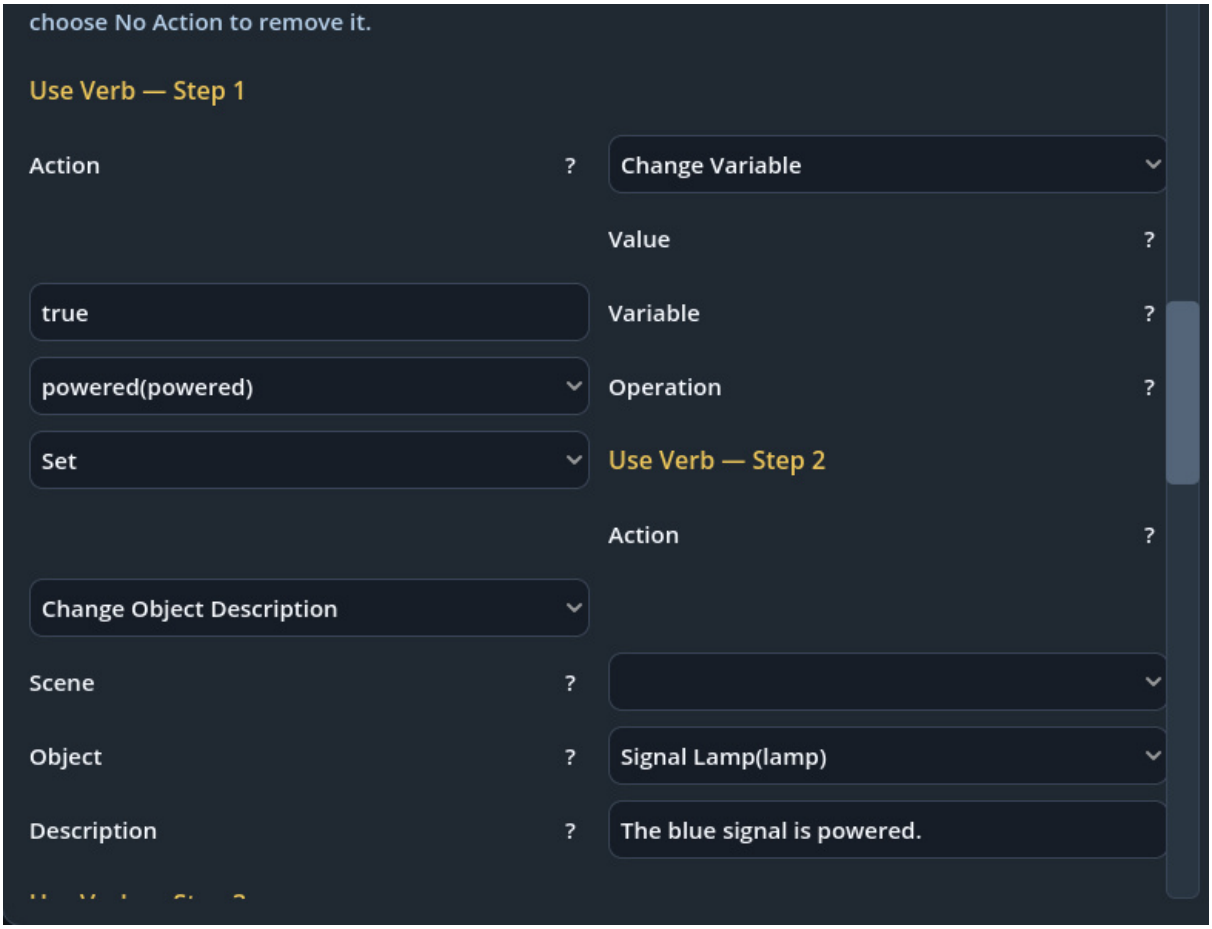

Figure 4. The native inspector uses Change Variable with Set to assign the powered flag.

Use a Boolean for an on/off fact. true, the number 1 and the text "true" are different kinds of value. Later, an If Variable node should compare powered with Boolean true.

> Set Variable is available in graphs, text wrappers and the shared GUI list, but not in this native inspector's current picker. Change Variable / Set performs the assignment here. You have not overlooked a hidden Set Variable button.

7. Add visible and audible feedback

Build the lever's response in an order that makes sense even if the player uses it more than once. Set a final state first; then present the feedback.

Step	Action	Fields
1	Change Variable	powered; Set; true.
2	Change Object Description	lamp; The blue signal is powered.
3	Play Sound	chime; channel puzzle.
4	Show Message	The signal is on.; 2.2 seconds.

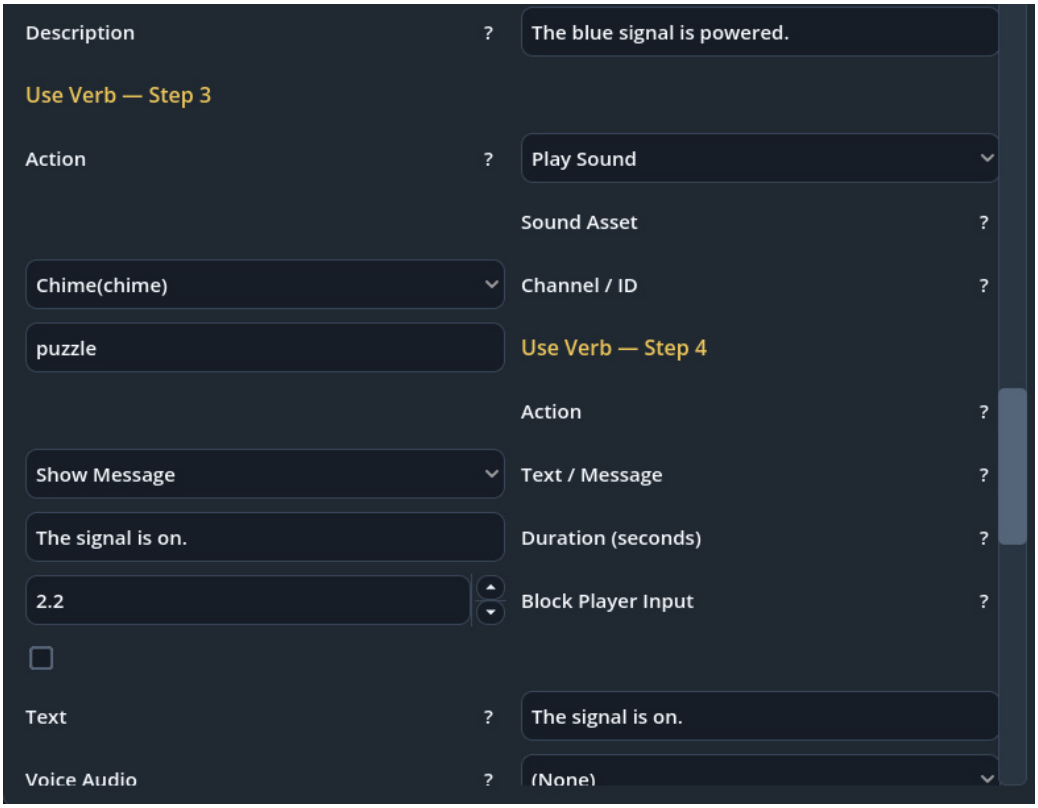

Figure 5. A later Use step starts a sound effect on the named puzzle channel.

The sound starts and the list continues to the message. The list does not wait for a sound effect to finish just because it appears earlier. If the timing needs a precisely placed pause, call a graph with Wait or use a cutscene timeline.

The lamp's new description is useful when its Look behavior uses the stored description. If you write a fixed Look message saying "The lamp is off," that fixed message will keep saying it until you change its logic too.

8. Test the whole response

1. Apply the inspector changes, save, then start Test Scene.
2. Choose Use and click Brass Lever. Confirm the message appears and the sound plays.
3. Use it again. powered should remain true, rather than toggling back to false.
4. Check the lamp's Look response. If it uses the description fallback, it should now describe the powered lamp.
5. Stop the preview before editing. Save the project again when the setup is correct.



Figure 6. The same signal message displayed by the running GUI event route later in the guide. Presentation is tested in Runtime Preview.

A useful test checks both the visible result and the underlying state. A message can display even when you changed the wrong variable. A variable can change even when the expected message is hidden under another interface.

> Test Scene starts a test session in the selected room. Also use Play / Restart for startup, room-entry order and main-menu behavior. GUI Designer Preview shows interaction appearance without running gameplay events.

9. Give the key to the player

The key on the workbench and the key in the bag are separate records. First create the Archive Key inventory definition, with ID key and an icon. Then give the visible scene object a Take response.

Step	Action	Fields
1	Give Inventory	Character: Player; Item: key; Quantity: 1.
2	Change Variable	key_taken; Set; true.
3	Change Variable	score; Add; 5.
4	Set Object Visible	Scene object: key_pickup; Visible off.

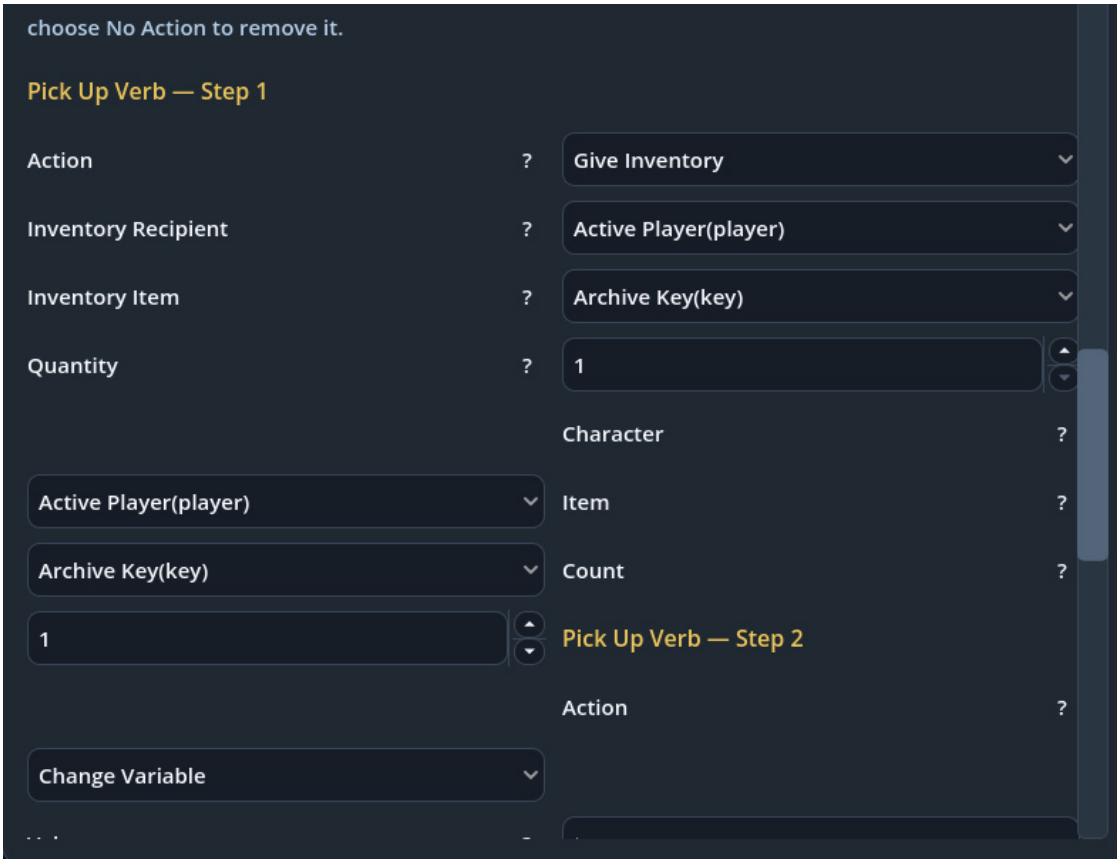

Figure 7. Give Inventory appears in the native Take response. Choose the carried item definition, not its scene mesh.

Apply, save and test. Choose Take, click the visible key, and open Inventory. The icon should appear, the world key should disappear, and score should rise by five.

> This simple sequence demonstrates the separate operations. Before treating it as a finished pickup, add the one-time guard in the next lessons. Otherwise another route could call the reward again.

10. Keep a one-time reward one-time

An event can happen more than once. Hover can happen whenever the pointer returns. Enter can happen whenever the player walks back in. A saved story flag is the dependable way to remember that a reward or greeting has already happened.

1. Create a visual script called Welcome, Once, with ID welcome_once.
2. Start with On Interact and connect Next to If Variable.
3. Select greeted, Equals true. Connect True to End.
4. Connect False to Set Variable: greeted = true.
5. Connect that node to Show Message: Welcome to the signal room. Set Duration to 2.2.

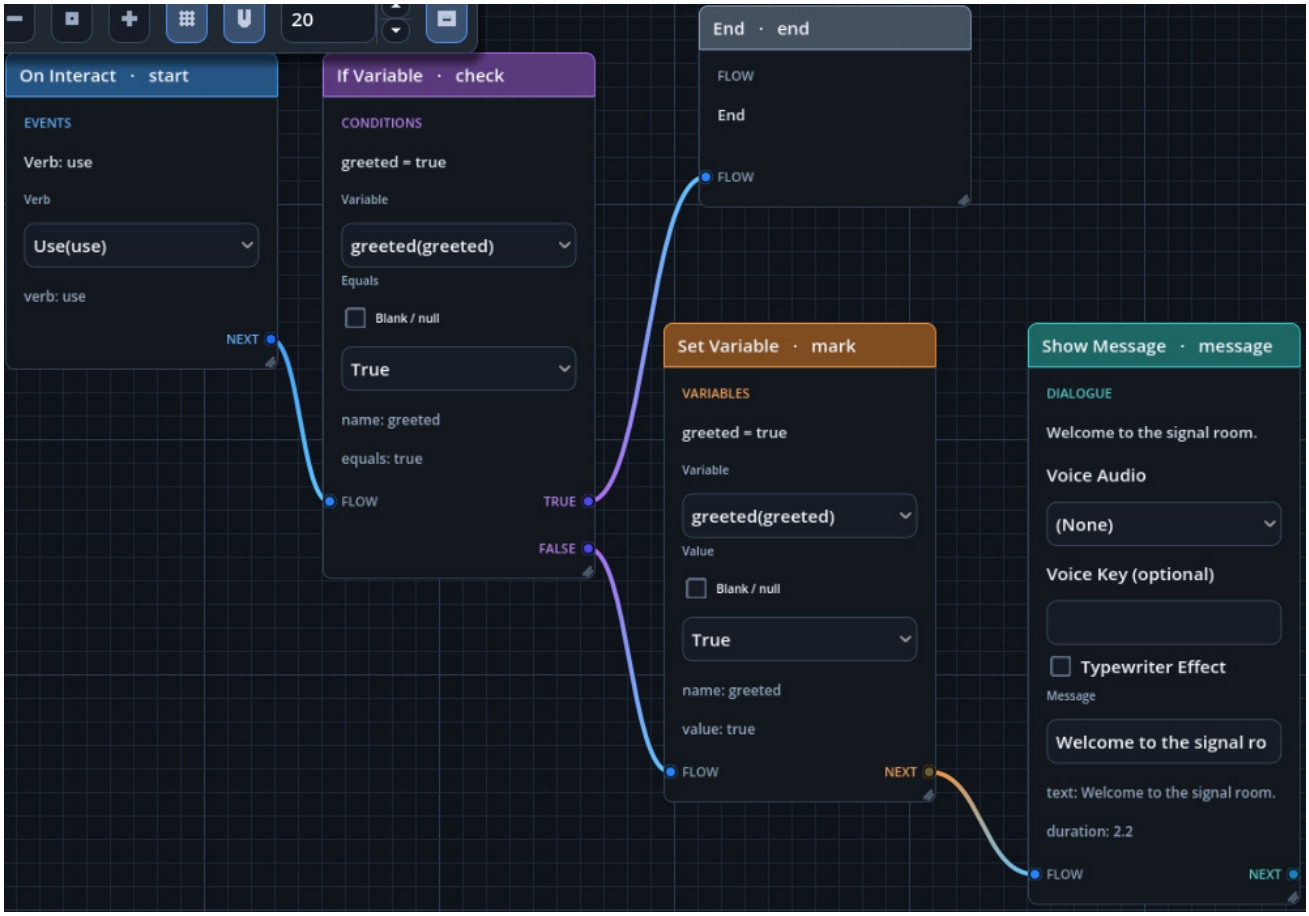

Figure 8. The True path ends; the False path marks the flag before showing the greeting.

Set the flag before a long message, sound or animation. That way, another entry route sees that the event is already underway. Adapt the same pattern to key_taken before the pickup reward.

> A trigger's temporary re-entry guard prevents overlapping execution of the same event while it is busy. It does not mean "only once in the whole game." That is the variable's job.

11. Attach the greeting to a trigger

1. Add a trigger volume in a reachable part of the room. Name it Welcome Area.
2. Open Trigger Events. Under Player Collision / Entry, choose Run Visual Script.
3. Select Welcome, Once. Apply Changes and save.
4. Start outside the volume in Test Scene, walk into it, walk out, and walk in again.
5. The first entry should show the greeting; later entries should reach the graph's End branch.

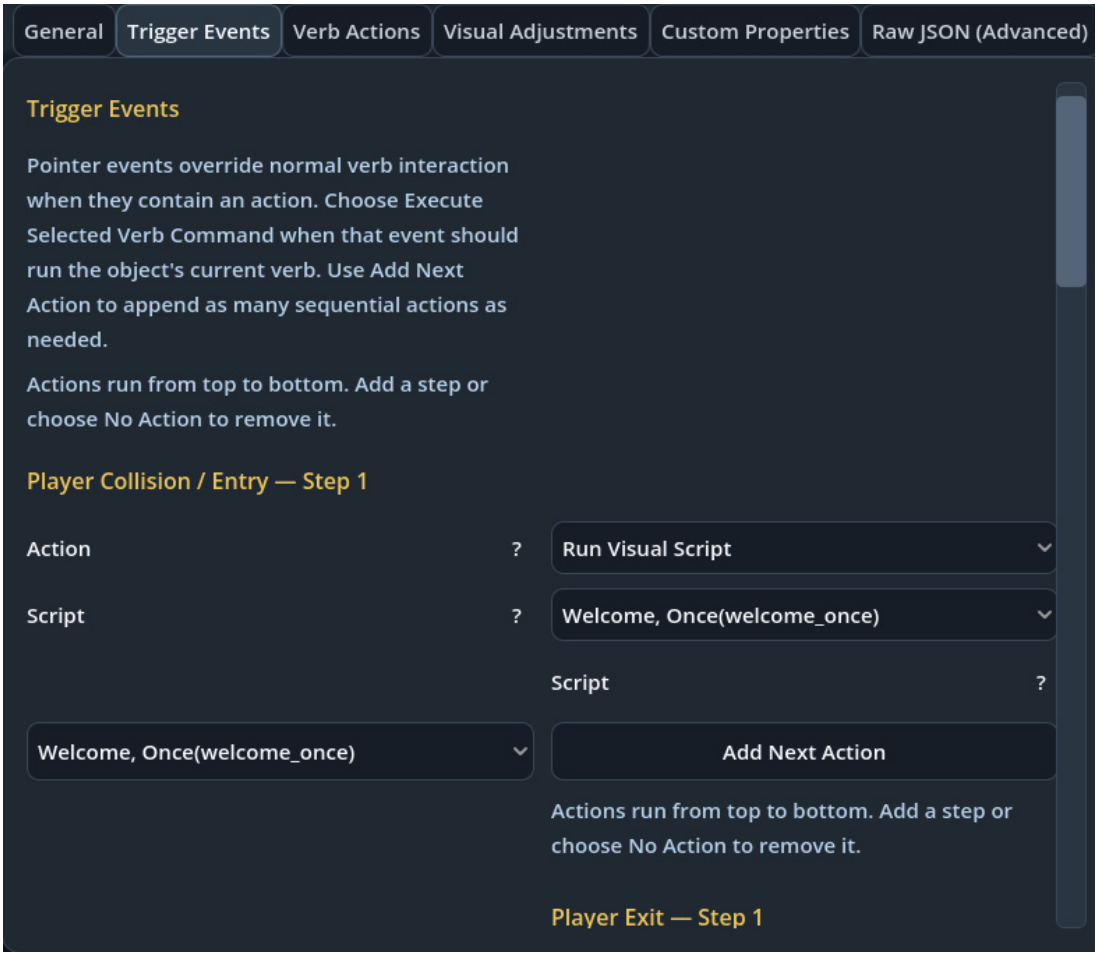

Figure 9. Player Collision / Entry calls the selected reusable script.

The checked 3D entry/exit path listens to the active player, not every NPC that happens to overlap the box. A trigger can also have direct click and verb responses, but those are separate event routes.

> Place the volume on the path the player actually walks, including the correct height. A visible editor outline is a guide to a volume; it is not proof that the player's collider crossed it. Test starting inside the volume as a separate case.

12. Open a door only with the key

Use a visual branch for the locked-door decision. A plain list containing Remove Inventory then Change Scene would change rooms even when the player had no key.

1. Create a graph named Open the Archive Door, ID open_door.
2. Connect On Interact to Has Inventory. Select Archive Key.
3. Connect False to Show Message: I need the archive key.
4. Connect True to Change Scene. Select The Archive and the from_signal spawn.
5. In the door's Use Verb response, choose Run Visual Script and select open_door.

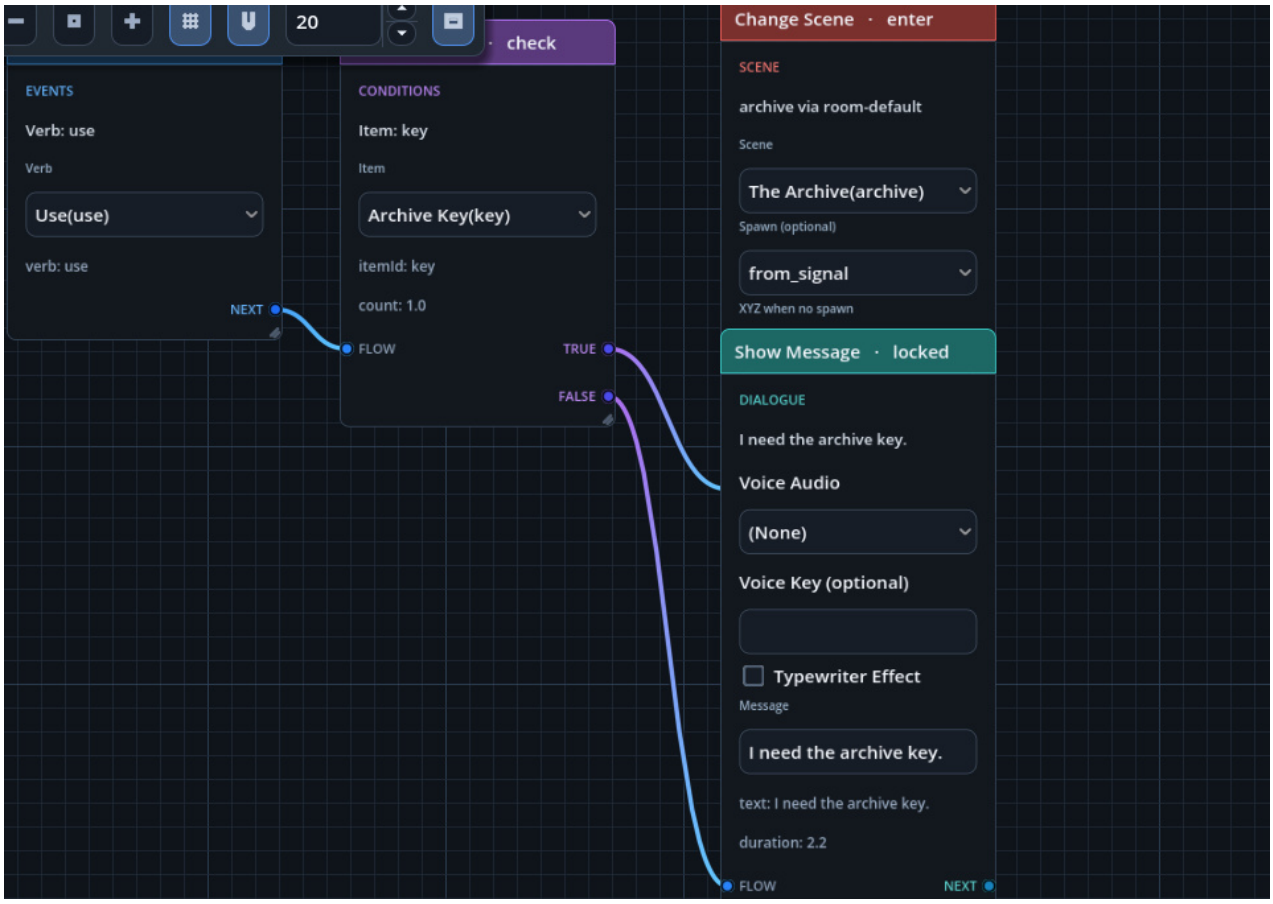

Figure 10. The inventory condition separates a useful failure response from the successful room change.

For a reusable key, leave it in the bag. For a consumed key, put Remove Inventory on the True path before changing rooms. In either case, the possession test comes first.

> Test both paths from a fresh session: first without the key, then after taking it. A successful run with a key does not prove the locked path works.

13. Entry and exit belong to the room

Select the scene record itself to edit Room Events. Selecting the floor or a wall opens that object's events instead. The room provides four local lists.

Visible label	Stored key	When it runs
When Enters Room	enter	Every normal room entry.
When Enters Room First Time	firstEnter	The first visit remembered in the current saved game state.
When Exits Room	leave	A normal player Change Scene departure.
When Exits Room First Time	firstLeave	The first such departure remembered in the current saved game state.

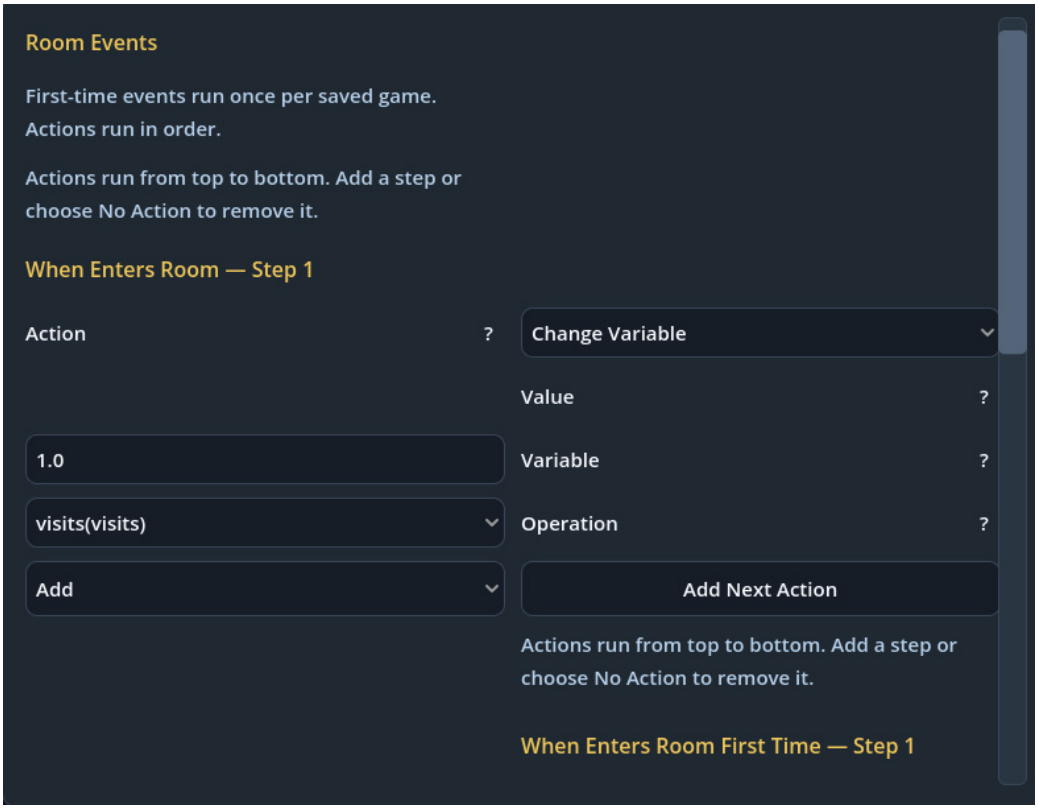

Figure 11. Room Events uses the same ordered native action fields as object responses.

For this room, use firstEnter to set the initial objective text, and enter to add 1 to visits. First-time is per saved game, not once per installation. A new game starts with fresh visit history; loading restores saved history.

> Do not put Create Variable or Set Variable resets in enter unless you really want them on every return. The player should not lose a completed puzzle just by walking out and back.

14. Understand room-event order

In the normal room-entry path, the runtime marks the room visited, runs firstEnter if appropriate, then runs enter. It emits OnSceneLoaded afterward. The visible inspector lists enter first, but that is not the execution order for a first visit.

Moment	Normal player room-change route
Leaving the old room	firstLeave if needed, then leave.
Beginning the load	OnSceneLeaving reports the old sceneId and nextSceneId.
Entering the new room	firstEnter if needed, then enter.
Entry lists are finished	OnSceneLoaded reports the new scene, previous scene and firstVisit.

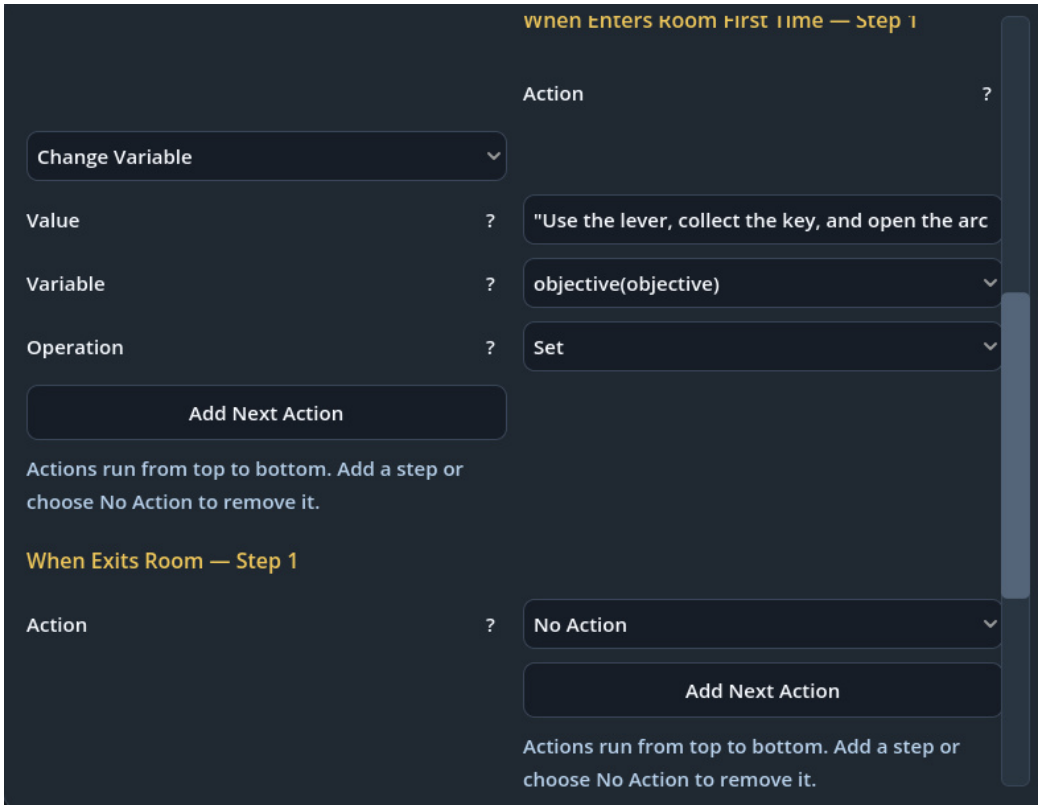

Figure 12. Use the first-entry list for the introduction or initial room objective.

This ordering matters when both a local list and a global listener change the same value. For example, set the objective locally, then let a global listener refresh a custom objective display from that value.

> Save loading and other low-level loading routes can have different needs from a normal player exit. Do not use a local leave event as the only cleanup for every possible load, restart, teleport or interrupted sequence.

15. Direct clicks and verbs are different routes

Object Events and Trigger Events contain direct pointer responses. Verb Actions contains the adventure responses such as Look and Use. A direct pointer list that contains work can override the normal verb interaction for that pointer event.

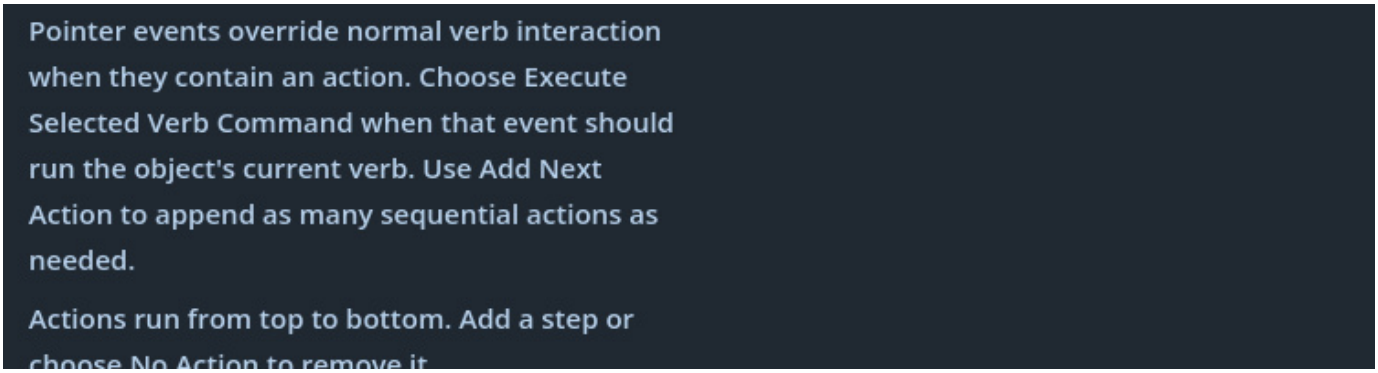

Figure 13. The editor explains the override at the top of Object Events.

You want...	Put the response here
Different results for Look, Use and Talk	The matching Verb Actions lists.
A click that always runs one special operation	The direct onClick response.
A trigger click to perform its currently selected verb	Execute Selected Verb Command in the trigger pointer response.
A hover label or brief highlight	Prefer presentation/pointer-feedback settings; keep hover logic lightweight.
The player to enter a volume	Player Collision / Entry, not a click response.

Double-clicks are not a promise that no single-click work occurred first. The first press of a double-click can already have reached a single-click route. Avoid destructive single-click actions on an element that also uses double-click for a different result.

> Do not place a long speech sequence on Hover. The player can re-enter the target, and a hovering pointer should not repeatedly lock the game while the player is trying to inspect the scene.

16. Give a GUI button a working response

For this build, use the convenience Run Visual Script behavior for a new multi-step button response. It can call a text script as well as a visual graph, and avoids the GUI catalog-action type-retention problem documented later.

```
func run(_context: Dictionary = {}) -> Variant:
    await ChangeVariable({"name":"powered",
                          "operation":"set", "value":true})
    await PlaySound({"assetId":"chime", "channel":"puzzle"})
    await ShowMessage({"text":"The signal is on.",
                       "duration":2.2})
    return true
```

1. Create a GDScript record named Power the Signal, ID power_signal. Enter the code above, validate, Apply Changes and save.
2. Select Send Signal in GUI Designer, then open Events.
3. Under Click choose the convenience run-visual-script behavior. Enter power_signal in Click Target.
4. Keep Additional Event Actions empty for this complete operation. Save and test in the running game.

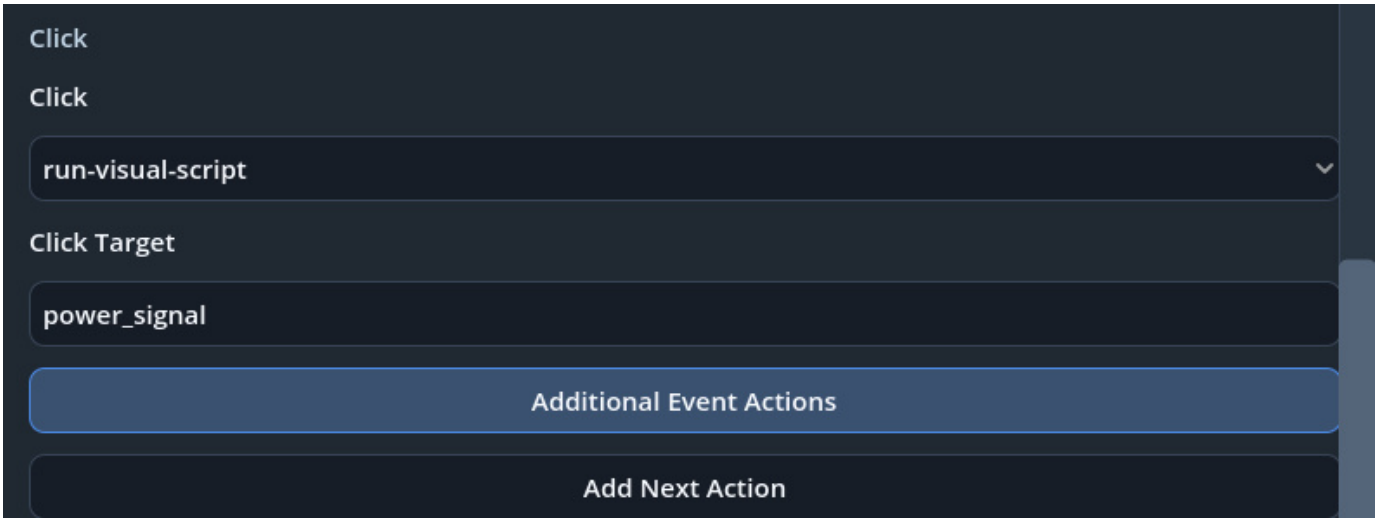

Figure 14. The convenience script behavior uses Click Target to identify the script.

The generic primary catalog action with a similar name is a different route. Use the convenience behavior shown here. In a fixed or pre-authored catalog action, the action fields are the inputs; the generic Target box is not their replacement.

17. Understand Additional Event Actions

Additional Event Actions is an ordered list attached to one GUI event. The runtime performs the primary event work first, then this list. Its row numbering starts again at Step 1; it does not count the primary behavior above it.

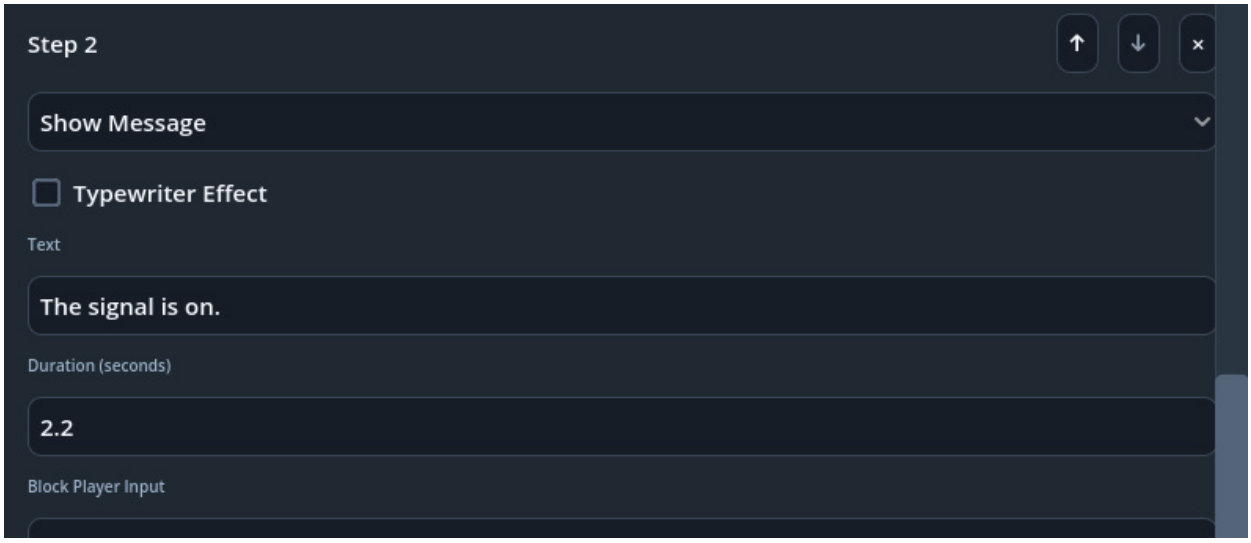

Figure 15. An existing valid additional list: its second row displays a message. The arrows move rows; the remove button deletes one.

A valid pre-authored arrangement can set powered in the primary action, then play a sound and display a message in two additional rows. That is an alternative to the complete power_signal script, not something to add after it: adding the same feedback again would make it happen twice.

When creating a new response in this checked build, keep the sequence inside power_signal and call it with the convenience run-visual-script behavior from Lesson 16. You can change its action order in the script or build a visual graph with the same steps.

Control	Purpose
Additional Event Actions	Expand/collapse the list.
Add Next Action	Add an additional row.
Up / Down	Reorder a row.
Remove button	Delete a row.
Action List primary behavior	An intended list-only arrangement; it still depends on valid additional action data.

The current shared editor loses the stored action type when a new additional row type is selected. Newly selected primary catalog actions have a related problem. Existing valid structured lists run, but the convenience script route is the dependable authoring method here.

18. Preview appearance; test behavior

GUI Designer Preview is for trying an element's appearance and built-in preview interaction. It intentionally does not run gameplay events. Hover/pressed images, slider movement and model-viewer manipulation can be previewed there without giving items, loading saves or changing rooms.

Check	Designer Preview	Test Scene / Play
Hover/pressed visual treatment	Yes, when configured on the element.	Yes, under runtime input.
A list that gives an item	No.	Yes.
A button calling a script	No gameplay execution.	Yes.
A dialogue or room transition	No.	Yes.
Bound text/variables	May show placeholder/design content.	Uses the live game state.



Figure 16. The running event has set Power to true and displayed its confirmation.

For a clean test, put the GUI on screen, click once, and check the state and the response. Pressed imagery should release back to the appropriate normal/hover state. That appearance is separate from whether a click action has run.

> If a designer click merely selects the element, that does not prove its event is broken. Selection is the editor's normal job outside preview mode.

19. Widget input is not a new global event

Every GUI element can use the common Hover, Click, Double Click, Right Click and Double Right Click routes where supported by its control. Widgets also have built-in jobs: toggles write their bound variable, sliders write a number, text boxes can submit parser commands, and save rows carry slot information.

Widget route	Useful data / behavior	Do not assume...
Check box / radio button	Updates the bound variable from its toggled state.	That there is a shared OnCheckBoxChanged subscription.
Slider	Updates its bound variable as the value changes.	That Click alone reports every drag value.
Text box	Can submit text to the parser or use its configured text behavior.	That every keystroke is a global action-list event.
Progress threshold	Provides percentage and value to its configured response.	That a threshold callback is one of the five mouse events.
Save-game row event	Adds saveSlot, saveExists, saveName and other slot metadata.	That the currently selected slot always exists.
Inventory item interaction	Uses the item's verb/combination actions and inventory context.	That an inventory icon is the same scene entity as the pickup.

For progress thresholds, keep one place responsible for ordered work. The threshold route can launch a selected script and an action list without a single shared await chain; put a strict sequence inside the script if its order matters.

> Use the widget's dedicated tutorial for its detailed layout and binding controls. Here the important distinction is which route starts your logic and which values that route actually passes.

20. Start a conversation from Talk

1. Create a dialogue record named Ask Elias. Give it an opening line, a choice node and an ending.
2. Select Elias in Scene Designer and open Verb Actions.
3. Under Talk, choose Start Dialogue and select Ask Elias.
4. Apply, save and test. Select Talk and click Elias.
5. Finish or cancel the conversation, then check that normal control returns.



Figure 17. The running conversation uses the character and dialogue presentation.

Start Dialogue waits for the conversation session. An action after it belongs to the end of that session, not the moment the first line appears. If it grants a reward, decide whether interruption should count as success.

Inside Dialogue Designer, an Action node runs a gameplay command between dialogue nodes. Connect it into the conversation flow. A disconnected action node does not run simply because it is visible in the graph.

> OnDialogueStarted and OnDialogueEnded are broader script notifications. Use them for shared camera/UI handling or logging, and filter the dialogue ID. Keep conversation-specific rewards in one deliberate branch.

21. Schedule overlapping work on a timeline

A cutscene event is scheduled at a time. That is a different meaning of event from a script subscription. Use the timeline when a camera blend, sound, animation and speech should overlap rather than form one blocking chain.

Time	Track	Action
0.0 s	Camera	Activate Camera: detail; blend 1.0; Wait off.
0.8 s	Audio	Play Sound: chime; channel puzzle.
1.0 s	Dialogue	Show Message: The signal is on.; duration 2.0.

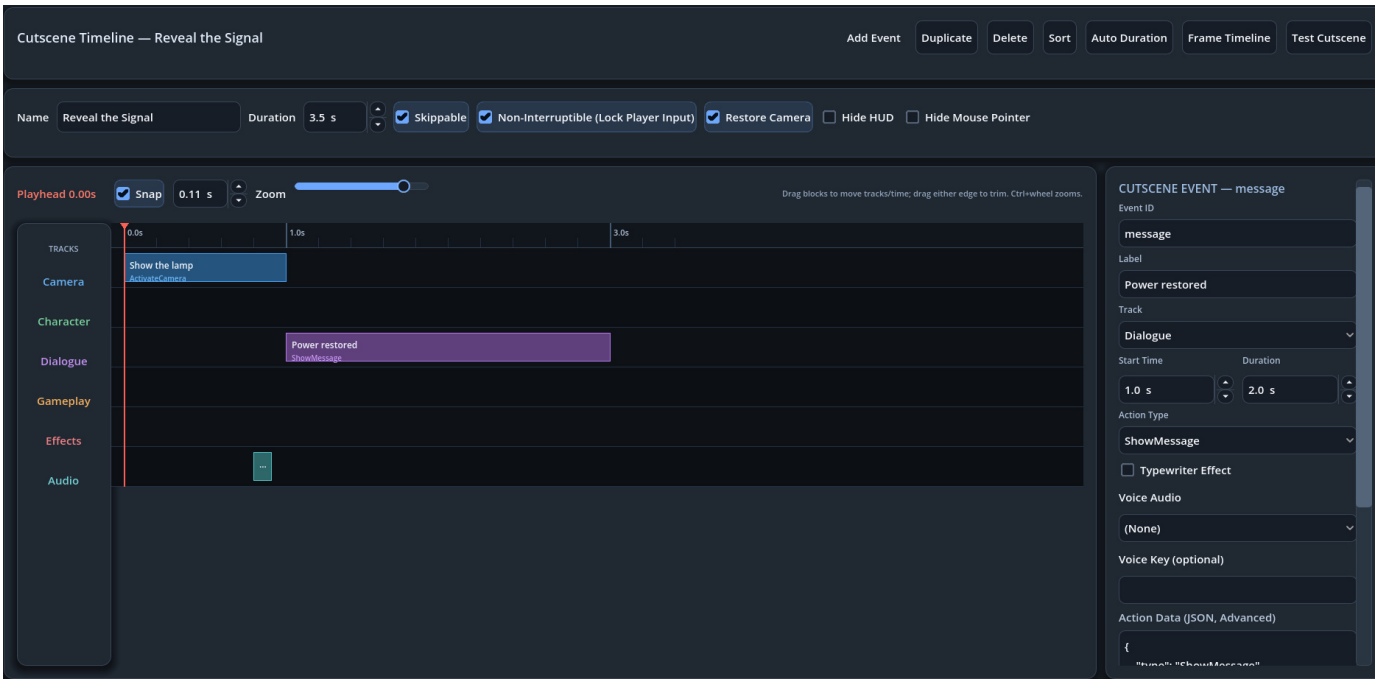

Figure 18. A selected timeline event has its own time/duration and Action Data.

The event block's duration is not a universal substitute for the action's own fields. Give a camera movement its intended duration and give a message its intended duration. Starting most imported animations returns after playback starts; it does not automatically hold the next action until the clip finishes.

> Put critical puzzle state outside optional cinematic decoration, or make the skip path apply that state too. A skipped camera reveal should not leave the door locked while the story says it is open.

22. Call the same actions from a script

A text script is useful when a short conditional operation is clearer in code. The supported scripting languages share the action names, while each language keeps its own syntax. The following is a complete GDScript example for the editor's standard PCA script base.

```
func run(_context: Dictionary = {}) -> Variant:
    await ChangeVariable({
        "name": "powered", "operation": "set", "value": true
    })
    await PlaySound({"assetId": "chime", "channel": "puzzle"})
    await ShowMessage({"text": "The signal is on.",
                       "duration": 2.2})
    return true
```

1. Create a GDScript record and give it a stable ID, such as power_signal.
2. Enter the code, validate it and click Apply Changes. Save the project.
3. On the lever or GUI event, choose Run Visual Script and select that text script.
4. Test the call through its real event, not just the editor's Run Script button.

Await means the caller waits for the action routine to return. It cannot force an action to wait for something the routine only starts. Play Sound starts playback; Show Message waits for its presentation routine. Animation timing needs the same distinction.

GDScript records without an explicit extends line receive PCA's standard script base. A custom base must inherit the PCA base to use these helpers. Godot's self still means the script instance; this means the calling game record.

23. Subscribe a script to a shared event

A shared event lets a script listen without being attached to one particular lever or button. For a simple test, count explicitly emitted custom events.

1. Create a numeric game variable custom_count starting at 0.
2. Create a GDScript named Event Counter, ID event_log.
3. In Event Functions, check OnCustomEvent. Keep global instance alive is optional for a counter stored in a game variable; it is useful when the script has its own persistent instance state.
4. Add the matching handler below, validate, Apply Changes and save.

```
func on_custom_event(event: Dictionary = {}) -> Variant:
    if event.get("reason", "") != "signal_powered":
        return false
    await ChangeVariable({"name": "custom_count",
                          "operation": "add", "value": 1})
    return true
```

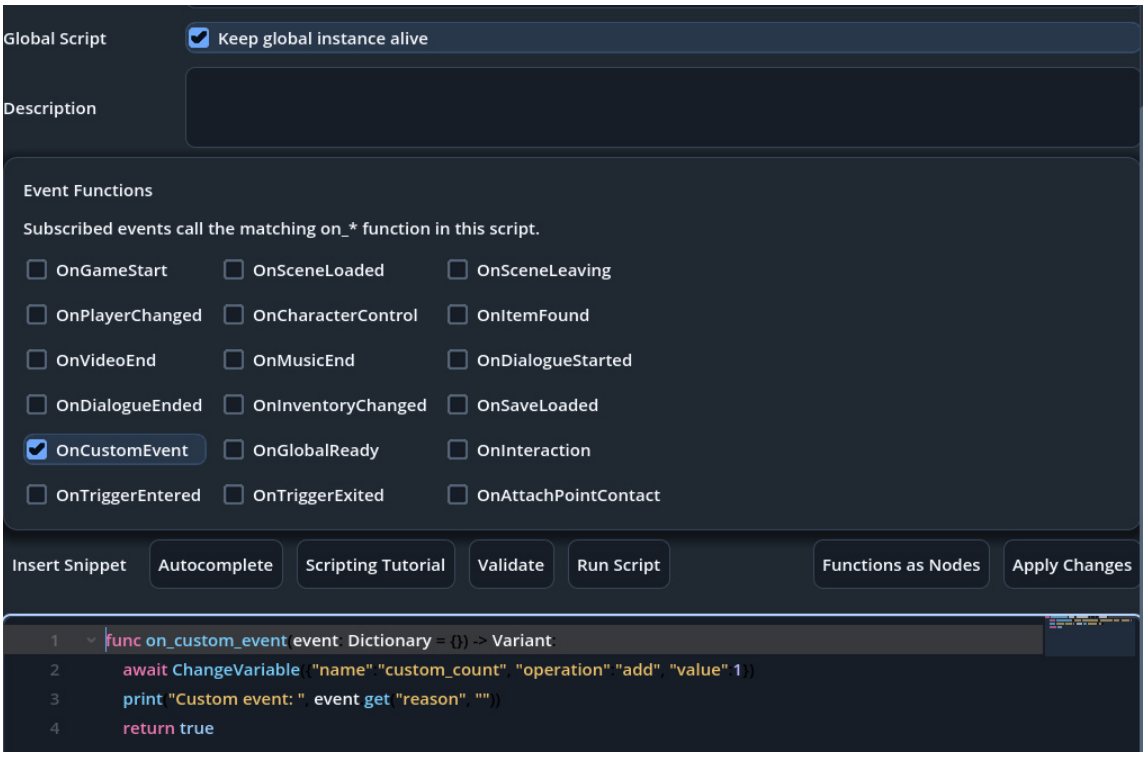

Figure 19. The subscription checkbox and matching on_custom_event function are both part of the setup.

A Global Script checkbox controls instance lifetime. It is not a shortcut for subscribing to every event. Likewise, writing on_custom_event without the subscription does not make this script a universal listener.

24. Emit and receive the custom event

The name OnCustomEvent does not fire by itself when a variable changes. An emitter must send it. Add an Emit Event node after the lever's successful state change, or use the helper in a text script.

```
await EmitEvent("OnCustomEvent", {
    "reason": "signal_powered",
    "sourceId": "lever"
})
```

For a visual listener, add On Event and select OnCustomEvent. Connect it to Read Context, with Context Key reason and Store in Variable last_reason. Then connect to End. This graph listens automatically; do not also call it from the lever unless you want an additional direct execution.

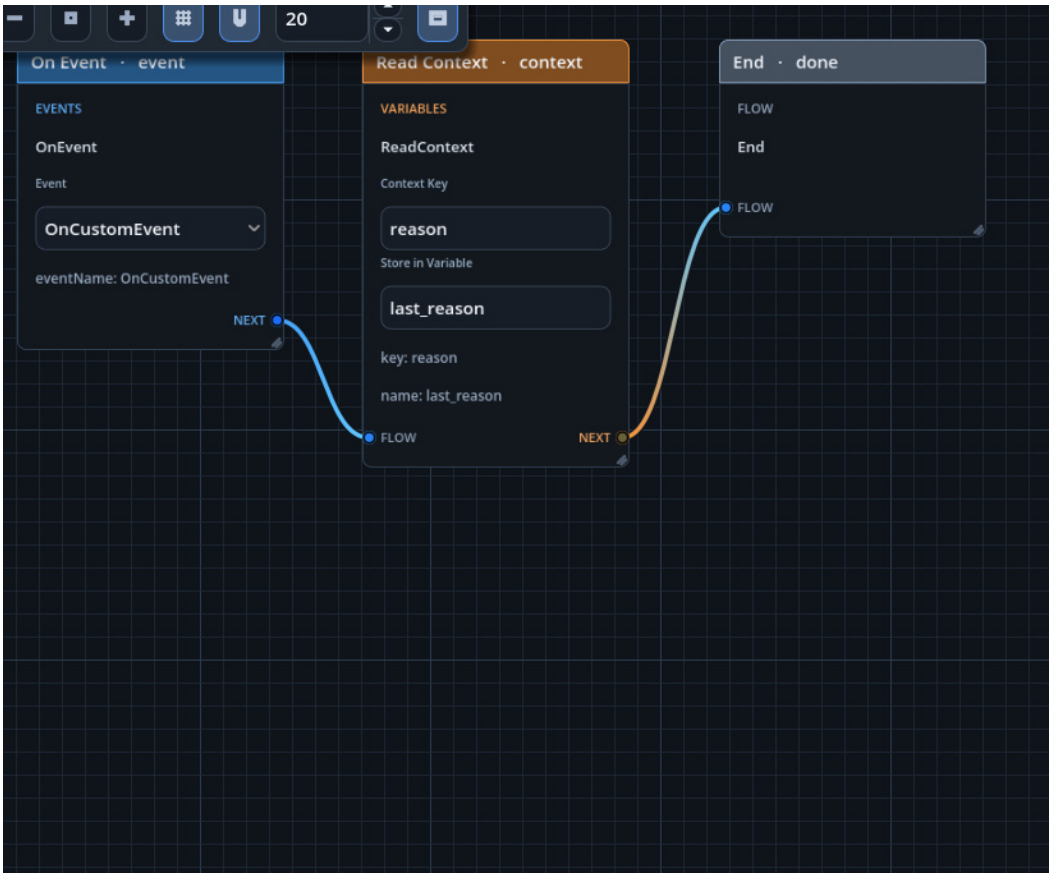

Figure 20. An On Event entry reads information from the custom event payload.

The checked dispatcher awaits the text-language subscription routes before its On Event visual graphs. Do not rely on the order of several unrelated listener records as a puzzle scheduler. Put dependent work in one list or graph.

> Do not emit the same event unconditionally from its own listener. It can call itself again. Filter the reason, use a guard variable, or emit a different notification that cannot loop back into the same work.

25. Use context instead of guessing

Context is information about this particular call. A trigger event can identify its trigger; a GUI event can identify the GUI and element; an inventory combination can identify the source and target item. Different event routes supply different keys.

Caller	Typical useful keys
Room local action list	sceneId, roomEvent.
Scene verb response	entityId, targetEntityId, targetKind, verb, inventoryItemId, sourceItemId.
Trigger local action list	entityId, triggerEvent; event/body data where supplied.
Shared trigger event	triggerId, entityId, sceneId, playerId, event; regionId for a region.
GUI element event	guiId, guiElementId, eventName such as GUI.click, guiEvent.
Inventory verb	itemId, inventoryItemId, verb.
Inventory combination	sourceItemId, targetItemId, inventoryItemId, targetKind.
Custom event	The dictionary sent by the emitter.

Use event.get("key", fallback) in a text handler when the key may be absent. In visual graphs, Read Context can copy a value into a game variable. An expression such as {"$context":"entityId"} reads the current call's value directly.

> The shared OnInteraction payload includes more keys than the local verb list that follows it. In this build the local list drops some shared scene/player/character keys. Code against the route you actually use, not against a combined imaginary payload.

26. Make this mean the caller

Use this to reuse one operation on several calling objects. A script called by the lever can treat this as the lever; the same script called by another prop can treat this as that prop. It does not automatically mean the player.

```
func run(_context: Dictionary = {}) -> Variant:
    if this == null:
        return false
    await SetObjectVisible({"entityId": "this",
                            "visible": false})
    return true
```

That example is a small "hide the calling object" operation. Use it only from an appropriate object-owned event. If an editor picker does not offer this, use the text script or structured expression route instead of choosing a random object as a substitute.

Situation	What to expect
Object/character/trigger event	A caller can normally be captured from its entity context.
GUI event	this identifies the owning GUI; elementId this has a special route for the calling element.
Inventory event	The item can be the owner when its item context is supplied.
Ownerless startup/room event	There may be no this record. Check for null.
Nested script call	The original captured caller is retained through the call chain.
OnCharacterControl	The controlled player can supply the character caller in this special event.

A structured action targeting this without a caller is skipped with a diagnostic in the checked runtime. It does not silently fall back to the player. The runtime check used for this guide confirmed that behavior.

27. Pass values, not look-alike text

Many action fields are typed. A checkbox is Boolean. A quantity is numeric. A general Value field can contain JSON. Keep the distinction between a literal value and an expression that fetches another value.

Entry / expression	Meaning
true / false	Boolean on/off values.
7 or 2.5	Numeric values.
"7" or "true"	Text strings, not the number or Boolean.
[0, 0, -2]	A three-number array used by vector fields.
{"$var":"score"}	Read the game variable score.
{"$context":"entityId"}	Read the current call's entityId.
{"$this":"id"}	Read the captured caller's ID.
{"$param":"points"}	Read a named parameter of a visual function.
{"$arg":0}	Read the first positional argument; indexing starts at zero.

A mapping such as {"$var":"score"} must be a JSON object in an expression/variant field. Pasting its characters into a plain text field may just create literal text. Use the Expression option where the editor supplies one.

> The runtime resolves these expressions recursively in action data. A missing variable, parameter or context key can resolve to null. It is better to validate an input near the start of a script than to let a later movement or reward action guess what null should mean.

28. Publish a function with editable parameters

Functions as Nodes turns existing script functions into selectable actions with a parameter form. It is useful when the same operation is needed with different values, such as awarding different numbers of points.

```
func reward(points: int, message: String) -> Variant:
    await AddScore({"amount": points})
    await ShowMessage({"text": message, "duration": 2.2})
    return true
```

1. Write and validate the function in a supported text script.
2. Open Functions as Nodes, then Add Function. Enter the exact function name reward and a friendly display name.
3. Add points as int and message as string, in the same order as the code. Choose sensible defaults.
4. Keep Expose as Action and Visual Node checked. Choose Use Definitions, then Apply Changes in the script editor and save.
5. In an action list or graph, choose Call Script Function, select the published function, and fill in its named parameters.

Use Definitions accepts the metadata in that dialog; it does not run the function or rewrite its source. If the function takes points and message, the definition must describe those parameters. A Result Variable can receive the returned true value, but an ordinary list will not automatically branch on it.

For visual functions, publish an existing entry node and use a Return node for a result. Named parameter expressions use $param. Removing the entry node or disabling exposure can invalidate an already authored function call.

Check the function definition

The definition below describes the reward function from the preceding lesson. Its names, order and types match the code: points first, then message.

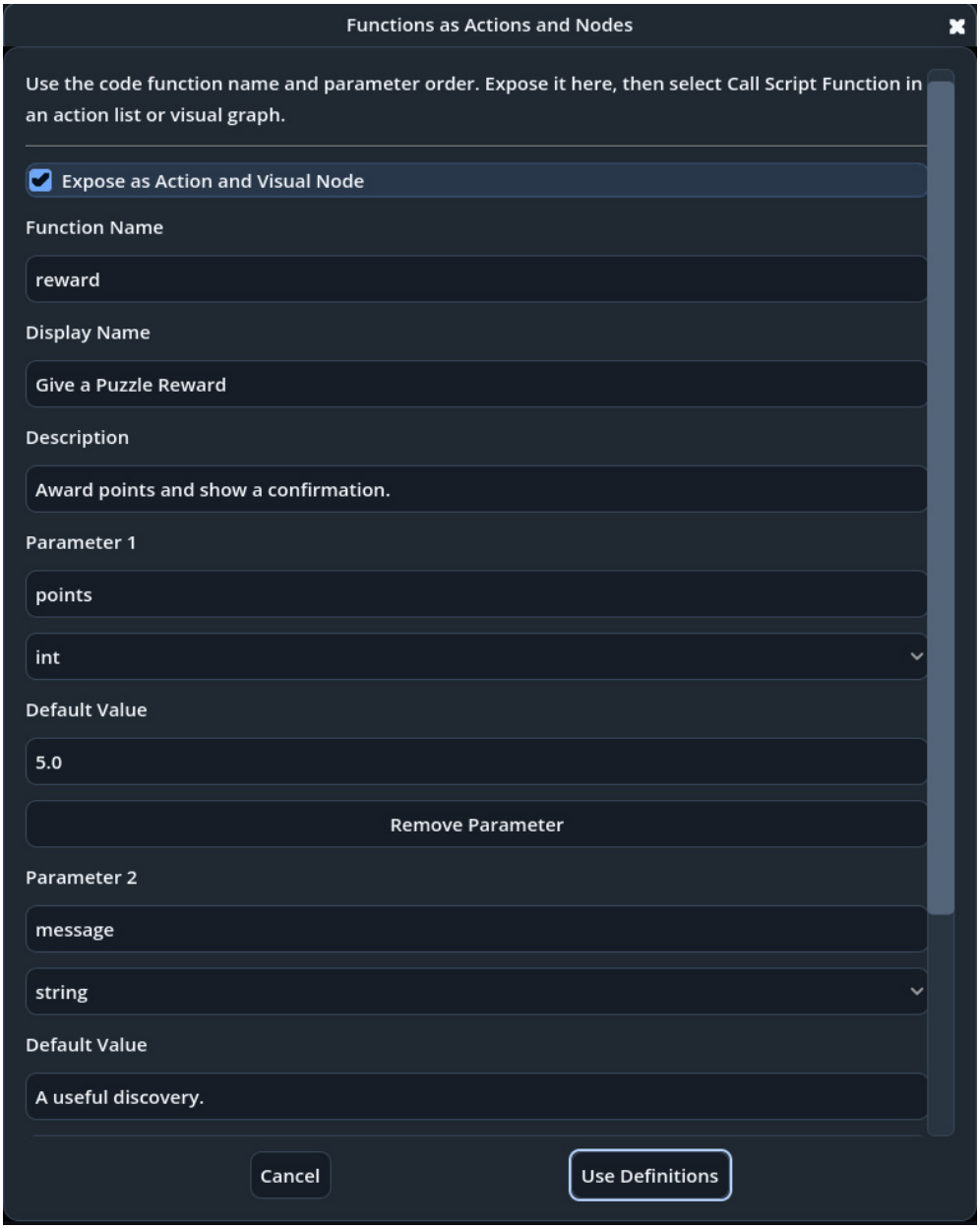

Figure 21. Functions as Actions and Nodes: the function metadata and Use Definitions button.

Use Definitions accepts this dialog. Apply Changes in the script editor then stores the script record, including the definitions. Save the project after both steps.

> If you change the definition later, keep the function signature in the source aligned. Metadata describes code; it does not rewrite the function for you.

29. Saving should preserve a finished state

Save after the important state changes agree with each other. For the key pickup, the bag, key_taken, score and world visibility should describe the same finished result. Saving halfway through an unguarded reward can make a loaded game inconsistent.

Action	Use it for
Save Game / Load Game	The basic route; without a structured slot it uses slot 0.
Save Game Slot / Load Game Slot	An explicit numbered slot and optional save name.
Save Game Widget Command	A custom save interface's selected slot or an explicit slot command.
OnSaveLoaded	Follow-up presentation after a successful state restore.

1. Save before taking the key, then take it and confirm the reward.
2. Save afterward in another slot.
3. Load the first slot. The key should be available and the reward should not yet be claimed.
4. Load the second slot. The bag should contain the key and the world pickup should remain hidden.
5. Try re-entering the room and retriggering any related response. Score should not rise again.

OnItemFound is not a lifetime achievement system. Its active-inventory route notices an item being added when it was absent. If the last copy is removed and later reacquired, that route can report another finding. Use a dedicated story flag for a permanent discovery reward.

> Treat slot overwrite and deletion as player decisions. Do not place a delete command in a hover response or use a broad global listener to erase saves as incidental cleanup.

30. Wait for the right thing

Action / route	What the next instruction can rely on
Set/Change Variable	The assignment/modification has been made.
Play Sound / Play Music	Playback has been started, not completed.
Show Message	The message routine has returned.
Say / Character Say	The speech routine has returned; Wait for Continue changes how it advances.
Play Voice	Waits for voice only when its Wait option is on.
Activate Camera / Zoom Blur	Wait behavior is controlled by the action's Wait option.
Play Animation	Most imported playback requests start and return; they do not wait the clip length.
Follow Character / Camera Orbit	A continuing behavior has started.
Fade Music / Fade Sound	The requested fade routine has returned.
Start Dialogue / Start Cutscene	The called session has returned.
Graph Wait	The chosen number of seconds has passed.

An await in a script waits for the routine, not for everything a routine happens to start. If you need a conversation line after a one-shot gesture, add an intentional delay or schedule both beats on a timeline. Do not make an endless animation or music loop responsible for signalling a natural end.

> For the legacy speech/message fields, values above 50 are treated as milliseconds by the compatibility helper. Use ordinary short second values, such as 2.2, in the current duration controls. Do not enter 60 expecting a one-minute message through that legacy route.

31. Filter completion events carefully

OnVideoEnd and OnMusicEnd are useful for natural completion. They are not universal notifications for every stop, seek, skip, replacement or loop. Test the exact route you intend to use.

```
func on_music_end(event: Dictionary = {}) -> Variant:
    if event.get("musicAssetId", "") != "opening_theme":
        return false
    await ShowMessage({"text": "The room falls quiet.",
                       "duration": 2.2})
    return true
```

1. Subscribe the script to OnMusicEnd.
2. Use a short, non-looping track for the first test.
3. Let it finish naturally, then repeat the test with Pause/Resume and with Stop.
4. Confirm that the response only runs when the intended completion route occurs.

Visual scripting also has dedicated On Video End and On Music End entry nodes with optional asset filtering. The broad On Event picker does not include those two names in this build. Avoid giving a graph two equivalent completion routes that each grant the same reward.

> A returned false value from a listener is not a global "cancel this event" instruction. Keep listeners independent and do not use undocumented return-value conventions as an event priority system.

32. Connect other designers to the same logic

Several designers can start a script or action route without creating a new shared subscription name. Reuse the same puzzle operation so keyboard, parser and mouse input do not grow separate versions of the game rules.

Source	Useful connection
Text Parser custom command	Matches configured command text and runs its selected script. Context includes source="text-parser", command, cleanedCommand and customCommand.
Normal parser verb/target	Can route into the same world/inventory interaction logic as the point-and-click interface.
Keybindings	Manual bindings dispatch their chosen command. Use the available command route and verify it in play; a held movement key is not a new OnFrame action-list subscription.
Character On Control hook	Runs when that character becomes controlled, alongside the shared control notification.
Inventory On Found hook	A dedicated item script hook in the active-item acquisition route. It has a different payload from the shared inventory event.
Quest completion actions	Run for the completed quest with questId context.
FMV markers / hotspots	Can run their configured action work with FMV timing context.

For example, let Use Door call open_door. If a parser command or custom GUI button also needs to open that door, call the same script with the appropriate target context. Keep the key check and successful room change inside that shared operation.

> Do not invent a shared event name just because a designer has a callback. The shared subscription reference lists the names actually exposed by this build.

33. Restore controls after a special mode

An inspection or cinematic mode often changes several independent things: player input, pointer visibility, movement profile, GUI visibility and camera ownership. Make a small restore plan before adding the effect.

Entering inspection	Leaving inspection
Disable player input if required.	Restore that input switch; other active locks can still apply.
Show the inspection GUI.	Hide the inspection GUI.
Activate the inspection camera.	Stop any orbit and activate the intended normal camera.
Hide or adapt the pointer deliberately.	Restore the intended pointer/device behavior.
Start any temporary sound/effect.	Stop the named channel/effect or let it finish safely.

Set Runtime Controls Enabled controls the control-profile feature as a whole. Set Player Input Enabled controls the player-input switch. Hiding the mouse pointer controls its presentation. Those commands do different jobs; they are not interchangeable locks.

Use explicit enabled/disabled states for cleanup. Toggle is convenient for a repeatable switch, but calling a cleanup toggle twice can leave the feature in the wrong state. Track temporary modes with a variable when several exits share the same restoration logic.

Test Close, Escape/back, scene change, dialogue interruption and cutscene skip where those exits exist. A sequence that restores control only on its happy path can leave the player stuck.

34. Watch an attach-point contact

Attach-point contact is useful when the important event is "these two named points meet" rather than "the player entered this large volume." A tool touching a socket is a good example.

1. Create a named point on the tool and a named socket point on the target.
2. Check their coordinate spaces, positions, enabled state and contact radii.
3. Start Watch Attach Point Contacts with a stable Watch ID, such as tool_touch.
4. Subscribe a script or On Event graph to OnAttachPointContact.
5. Filter watchId and phase. On the first valid enter, set your success flag before rewarding the player.
6. Stop the watch when the puzzle phase ends or the tool is put away.

```
func on_attach_point_contact(event: Dictionary = {}) -> Variant:
    if event.get("watchId", "") != "tool_touch":
        return false
    if event.get("phase", "") != "enter":
        return false
    if Runtime.variables.get("socket_done", false):
        return false
    await SetVariable({"name": "socket_done", "value": true})
    await StopAttachPointWatch({"watchId": "tool_touch"})
    return true
```

Swept detection can catch contact that happens between samples. It can report a quick enter and exit in a short interval. That is why the phase filter and one-time success flag matter.

Attach Points Touching returns a value; If Attach Points Touching branches a graph; Watch Attach Point Contacts creates a continuing watch. Choose the one that matches the job.

35. Build one dependable puzzle operation

A finished interaction should have a clear entrance, a condition, a state change, feedback and an exit. Use this checklist to turn the simple key-and-door lesson into a reliable game sequence.

Operation	Recommended order
Take key	Check key_taken. If false, set it true, give the item, update score, hide/disable the pickup, then present feedback.
Use lever	Check or set powered explicitly. Update any dependent description/effect. Play feedback. Do not toggle accidentally on repeated calls.
Use door without key	Give a useful locked response. Leave the room and inventory unchanged.
Use door with key	Apply any permanent unlock flag. Consume the key only if intended. Change to the destination spawn.
Return to room	Read saved flags and restore the matching presentation. Do not reset puzzle facts.
Load save	Use restored state. Refresh custom presentation through the successful-load path if needed.

When the player can reach the same operation by mouse, parser, controller or GUI, share the operation. The input routes can be different; the actual rule that decides whether the door opens should be the same.

> If a reward depends on several actions succeeding, write that dependency explicitly in a script. The ordinary list runner does not inspect an action's returned error and automatically roll the earlier changes back.

36. Diagnose a response that does nothing

Symptom	Check first
Nothing happens in GUI Designer	Gameplay events are not executed by designer preview. Use Test Scene.
Wrong verb response runs	Interaction mode, Default Verb, selected verb and direct pointer overrides.
A GUI button only selects a verb	The event's separate Verb setting has priority over its primary behavior.
Trigger never fires	Volume position/height, enabled state, active player overlap and starting position.
Trigger repeats a reward	A saved completion flag; temporary busy/re-entry guards are not once-ever flags.
Door opens without a key	A real condition branch before Change Scene; removing an item is not the check.
A newly selected GUI action does nothing	The GUI catalog type-retention issue in this build. Use the convenience run-visual-script behavior.
Change Scene has no room picker	That shared form lacks the specialized field. Put Change Scene in a visual/text script.
Animation starts but next step runs	Most imported animation calls return after starting. Add deliberate timing.
Message or dialogue never continues	Wait for Continue, visible Continue controls, voice pause and modal UI/input locks.
this does nothing	Whether the event has a valid captured caller; use explicit IDs for ownerless work.
Object property changes in data only	Whether that property has a live update path; use the dedicated action where possible.
After a scene change a target is missing	The old entity may no longer be loaded. Put room-local work before the transition.

Reduce the failing response to one Show Message. If that works, restore one action at a time. Check each target ID and typed value. A four-step test is easier to understand than a twenty-step test with three different failures.

37. Test the awkward cases too

A puzzle working once is a good start. The following checks catch the mistakes that players find by doing perfectly reasonable things in a different order.

Test	Expected result
Fresh new game	Starting variables, room state and initial control agree.
Test Scene directly	The room is usable even when the normal introduction is bypassed.
Use before examining	A missing Look step does not break the puzzle unless it is an explicit requirement.
Use without required item	Useful failure feedback; no item/reward/room mutation.
Use twice quickly	No duplicate rewards or nested conversations.
Leave and re-enter trigger	Only the intended repeatable response repeats.
Start inside trigger	Startup overlap behavior is understood.
Double-click a single-click target	No unintended double reward or destructive action.
Change player	Correct bag, caller, HUD and control setup.
Save before / after; reload both	World and inventory match the saved story state.
Skip or cancel cinematic	Critical puzzle state and input/camera cleanup remain correct.
Different input device	The intended verb/coin/proximity route still reaches the same rules.
Exported build	The target device can resolve assets, media and writable save paths.

The examples were checked against the native editor and runtime, including ordered values, variable creation, failure continuation, raw conditional execution, custom-event dispatch, caller resolution and the one-time graph. That does not replace testing your own room, assets and target hardware.

Shared script events: the full reference

The GDScript, PCA Script, Lua and BASIC script editors expose 18 shared subscription names. A handler runs because the script is subscribed or because a dedicated hook explicitly calls it; writing a similarly named function alone is not a universal subscription.

The payload lists below describe the standard emitters in this build. A context is a dictionary of information supplied by the caller, not a fixed global object. Local actions, inherited calls and language-specific initialization can add or omit keys.

> The visual On Event picker exposes 15 of these names. Video and music have dedicated graph entry nodes; Global Ready belongs to script initialization. Read the media and ready entries before assuming identical support on every surface.

OnGameStart

GDScript handler: `on_game_start(event: Dictionary = {})`

The runtime reaches its game-start hook. It is a game-session event, not a once-per-room event.

Standard context	Practical use / limits
sceneId, playerId	Initialize session-level logic here. Do not reset saved puzzle variables on every scene entry. A direct Test Scene can start a test session too; test a normal new game and a loaded game separately.

OnSceneLoaded

GDScript handler: `on_scene_loaded(event: Dictionary = {})`

A room has been constructed and its local entry actions have run.

Standard context	Practical use / limits
sceneId, previousSceneId, playerId, firstVisit	Filter sceneId before changing room-specific props. In the normal entry path, firstEnter runs before enter, then this event is emitted. It is not a substitute for the room's first-time event.

OnSceneLeaving

GDScript handler: `on_scene_leaving(event: Dictionary = {})`

The scene-loading route is leaving one scene for another.

Standard context	Practical use / limits
sceneId, nextSceneId	sceneId is the departing room. Do short cleanup here. The normal player Change Scene route also runs its local firstLeave/leave action lists; other loading routes need not reproduce every local transition hook.

OnPlayerChanged

GDScript handler: `on_player_changed(event: Dictionary = {})`

The controlled playable character changes.

Standard context	Practical use / limits
playerId, displayName	Refresh character-specific presentation. This event concerns player selection, not every movement of a character. The runtime follows it with OnCharacterControl in the switch path.

OnCharacterControl

GDScript handler: `on_character_control(event: Dictionary = {})`

A character becomes the controlled character, including initial control at startup.

Standard context	Practical use / limits
playerId, displayName	Use it for HUD/inventory presentation or control-specific setup. It also has a per-character On Control script hook. This event is a special case in which playerId can identify the calling character for this.

OnItemFound

GDScript handler: `on_item_found(event: Dictionary = {})`

The active-inventory add route adds an item that was previously absent from that bag.

Standard context	Practical use / limits
characterId, itemId, amount	It is based on previous possession, not a permanent lifetime discovery log. Remove the last copy and acquire it again, and this route can report it again. Not every NPC/character transfer route emits this event; use a story flag for one-time discoveries.

OnVideoEnd

GDScript handler: `on_video_end(event: Dictionary = {})`

The runtime receives the video-finished callback.

Standard context	Practical use / limits
videoAssetId	Filter the asset ID. Do not assume an explicit Stop Video or an interrupted/looping clip has the same natural-end behavior. Dedicated On Video End graph nodes are an additional entry route; avoid duplicating the same reward in both routes.

OnMusicEnd

GDScript handler: `on_music_end(event: Dictionary = {})`

The music-finished callback reaches the runtime.

Standard context	Practical use / limits
musicAssetId	Use a non-looping track when testing. Stopping or replacing a track is not the same as natural completion. Dedicated On Music End graph nodes can also listen for a selected asset.

OnDialogueStarted

GDScript handler: `on_dialogue_started(event: Dictionary = {})`

A valid dialogue session begins, before its graph is processed.

Standard context	Practical use / limits
dialogueId, playerId; inherited caller context may also be present	Filter dialogueId before adjusting a conversation camera or HUD. Starting the same dialogue from this handler can recurse. Keep listener work short when it must complete before the first line appears.

OnDialogueEnded

GDScript handler: `on_dialogue_ended(event: Dictionary = {})`

A dialogue session ends, including interrupted endings.

Standard context	Practical use / limits
dialogueId, playerId, interrupted; inherited caller context may also be present	Check interrupted before granting a reward that belongs only to successful completion. Some explicit end paths can supply a blank dialogueId, so do not assume every payload identifies a normal completed graph.

OnInventoryChanged

GDScript handler: `on_inventory_changed(event: Dictionary = {})`

An inventory mutation route reports an add, remove or quantity change.

Standard context	Practical use / limits
characterId, itemId, amount, operation	Use operation and the actual bag state together. amount is supplied by that mutation route; it is not guaranteed to be the resulting total. Adding/removing inside this listener can trigger another inventory event, so guard feedback loops.

OnSaveLoaded

GDScript handler: `on_save_loaded(event: Dictionary = {})`

A save has successfully restored game state.

Standard context	Practical use / limits
slot, sceneId, playerId	Refresh presentation from restored variables. Do not award a pickup again merely because its saved state was loaded. This is a successful-load hook; a missing or invalid save should not be treated as a completed restore.

OnCustomEvent

GDScript handler: `on_custom_event(event: Dictionary = {})`

A script or Emit Event graph node explicitly emits this event name.

Standard context	Practical use / limits
Whatever keys the emitter sends; eventName is added for the listener	Use a reason key, such as signal_powered, to distinguish several messages sharing OnCustomEvent. It is not automatically emitted by every button or variable edit. The graph form uses its Context JSON; the GDScript helper merges its current caller context before emitting.

OnGlobalReady

GDScript handler: `on_global_ready(event: Dictionary = {})`

A persistent/global script is initialized through its language runtime's ready route.

Standard context	Practical use / limits
Language-specific startup context; native GDScript includes eventName and scriptId	Use it to initialize that global instance, not to assume a playable room is ready. The GDScript runtime calls a global script's matching method directly. The visual On Event picker does not offer this ready event; do not design it as a universal room-ready graph broadcast.

OnInteraction

GDScript handler: `on_interaction(event: Dictionary = {})`

A scene-entity verb interaction reaches its execution point.

Standard context	Practical use / limits
entityId, targetEntityId, targetKind, sceneId, playerId, characterId, verb, inventoryItemId, sourceItemId; regionId for a region	This is the shared notification before the local verb action list. Filter both target and verb. The local list receives a related but smaller context; do not assume every key shown here is passed unchanged to every nested call.

OnTriggerEntered

GDScript handler: `on_trigger_entered(event: Dictionary = {})`

The active player enters an eligible trigger/interactive-region overlap.

Standard context	Practical use / limits
triggerId, entityId, sceneId, playerId, event="enter"; regionId for a region	In the checked 3D route, other NPC bodies do not trigger this player-entry hook. The shared event runs before local onEnter actions. Starting inside a volume can create startup contact behavior; test that case separately.

OnTriggerExited

GDScript handler: `on_trigger_exited(event: Dictionary = {})`

The active player leaves an eligible trigger/interactive-region overlap.

Standard context	Practical use / limits
triggerId, entityId, sceneId, playerId, event="exit"; regionId for a region	Use it for leaving a local effect area. Teleports and room changes are not guaranteed to behave like ordinary walking out of the same live volume; perform important cleanup in the owning scene/sequence too.

OnAttachPointContact

GDScript handler: `on_attach_point_contact(event: Dictionary = {})`

A named attach-point watch reports entering or leaving point contact.

Standard context	Practical use / limits
watchId, phase, entityId, point, targetId, targetPoint, sceneId, touching, swept, position, distance, fraction, contactRadius, dimension	Filter watchId and phase="enter" for a one-shot success. A swept contact may report enter and exit between samples. Stop the watch when its job ends and set a completion flag before awarding the result.

GUI primary behaviors

These convenience behaviors appear alongside catalog actions in an element's Events tab. Their Target box is not the same as every catalog action's own target fields. When a catalog action is selected, fill in its generated fields instead.

Behavior ID(s)	Meaning / target
none	No primary action. Additional Event Actions can still run. A legacy click Action string can still be used by the fallback route.
action-list	Use the Additional Event Actions list as the event's work. There is no separate primary behavior to execute.
toggle-gui / show-gui / hide-gui	Target is the authored GUI ID. Toggle reverses visibility; Show/Hide request a specific state.
run-visual-script	Target is a visual or supported text-script ID. It receives GUI event context.
set-verb	Target selects a verb. The separate event Verb selection also has verb-selection behavior; avoid combining two conflicting choices.
switch-character	Target is a playable character ID. Its playable/switch settings still apply.
toggle-inventory / toggle-journal	Open/close the runtime inventory or journal route, using the configured system interfaces.
inventory-previous / inventory-next / inventory-up / inventory-down	Navigate the intended inventory widget. Target identifies the configured widget route; test the correct inventory owner and scroll mode.
save-widget-previous / save-widget-next / save-widget-up / save-widget-down	Navigate a save-game widget.
save-widget-load / save-widget-save	Load or save the selected slot through the target save widget.
save-widget-rename / save-widget-delete / save-widget-refresh	Rename/delete the selected save, or refresh the widget. Treat deletion as a deliberate player operation.
dialogue-continue	Continue the active speech/dialogue presentation. Use it on a visible Continue control.
system-resume	Close the system interfaces and return to play.
system-main-menu	Open the game's main-menu route.
system-save / system-load / system-options	Open the configured system save, load or options interface.

Behavior ID(s)	Meaning / target
system-new-game	Start a new game through the system-menu route.
system-continue-game	Use the system continuation route for a saved game.
system-quit	Request the game's quit route. Browser/mobile behavior differs from a desktop process exit.
custom	Interpret the Target as a legacy action string, such as a script or GUI command. Prefer the explicit modern behavior when one exists.

A selected event Verb takes the verb-selection route before the primary behavior. Additional Event Actions are still processed afterward. Leave the event Verb blank when the button is meant to run a direct primary action instead.

Graph entries, conditions and flow nodes

These nodes explain why a visual graph can make decisions that a plain ordered list cannot. They are reference material for connecting action lists to visual scripts, rather than extra instructions that can all be dropped into every list picker.

True/False are branch ports; Next continues ordinary flow. A graph starts at its selected entry and follows connected wires. A disconnected action is not an instruction scheduled to run later.

OnInteract

An entry node for a directly called interaction graph. Its verb setting helps choose/filter the entry.

Use one at the start of open_door. It is not a global subscription to every object interaction; On Event / OnInteraction is the broad listener.

Field / stored key	Input	Default where specified
Verb `verb`	Choose a Verb Designer ID.	-

OnVideoEnd

A dedicated video-completion entry, optionally filtered by an asset.

Avoid also giving the same graph an equivalent broad On Event listener unless you intend two routes.

Field / stored key	Input	Default where specified
Video `assetId`	Choose an imported video.	-

OnMusicEnd

A dedicated music-completion entry, optionally filtered by an asset.

Use a non-looping track to test natural completion.

Field / stored key	Input	Default where specified
Music `assetId`	Choose an imported audio asset.	-

HasInventory

A True/False condition for possession of an inventory item.

Use the item ID; branch False to a useful response and True to the successful action sequence.

Field / stored key	Input	Default where specified
Inventory Item `itemId`	Choose an inventory definition ID.	-

IfCurrentRoom

A True/False test for the current scene.

Prefer it to the value-returning Is Current Room action when building visible graph branches.

Field / stored key	Input	Default where specified
Scene `sceneId`	Choose a scene ID.	-

IfAttachPointsTouching

A True/False test of the selected attach-point pair and tolerance.

Use it for a sampled check; a Watch plus OnAttachPointContact handles contact over time.

Field / stored key	Input	Default where specified
Object / Character `entityId`	Choose a scene entity ID.	-
Attach Point `point`	Name of the source attach point.	-
Target Object `targetId`	Choose a scene entity ID.	-
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	-
Contact Tolerance `tolerance`	Extra allowed contact distance; use a nonnegative number.	0.0

IfPlayerWasHere

A condition based on recorded character/room visit history.

Use it for a returning-player observation, rather than a general test that a room file exists.

Field / stored key	Input	Default where specified
Scene `sceneId`	Choose a scene ID.	-

IfPlayerWasNeverHere

The complementary visit-history condition.

Use it for a first-visit response. A local firstEnter room event is often simpler for one room's introduction.

Field / stored key	Input	Default where specified
Scene `sceneId`	Choose a scene ID.	-

IfVariable

Compare a variable with an Equals value and follow True or False.

Use Boolean true for greeted, rather than the text "true". A raw nested-action form also exists, but the ordinary list picker is not a visual branch editor.

Field / stored key	Input	Default where specified
Variable `name`	Choose a game variable name.	-
Equals `equals`	A typed value or valid JSON.	-

IfInventoryItemWasUsed

Check the inventory item passed in the interaction context.

It asks what was used in this call, not merely whether the bag contains the item.

Field / stored key	Input	Default where specified
Inventory Item `itemId`	Choose an inventory definition ID.	-

IfCharacterMoving

Branch on whether a character is moving.

Use it to postpone a staged pose or give a different observation; it does not itself wait for movement to end.

Field / stored key	Input	Default where specified
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	-

WhileLoop

Repeat its body while a variable comparison is true, up to a safety limit.

Change the tested value inside the loop. A safety limit prevents a runaway graph, but does not make an incorrectly designed loop useful.

Field / stored key	Input	Default where specified
Variable `name`	Choose a game variable name.	-
Operator `operator`	Choose: ==, !=, <, <=, >, >=.	-
Value `value`	A value of the type required by the selected variable, operation or property.	-
Safety Limit `maxIterations`	Whole number.	-

ForLoop

Repeat its body a set number of times and expose an index variable.

Use small counts first. Check the visible Body/Done flow rather than drawing an unbounded wire back to the start.

Field / stored key	Input	Default where specified
Iterations `count`	Number of copies or repetitions. Use a whole number.	-
Index Variable `indexVariable`	Text.	-

EndVisualScript

Request visual-script termination, optionally extending to the calling graph.

It is also catalogued as an action. It does not break an unrelated native/GUI array runner.

Field / stored key	Input	Default where specified
Also End Calling Script `endParent`	Also request termination of the calling visual graph.	-

End

A graph terminator.

Use it on an already-complete branch, such as greeted already true.

OnEvent

An automatic graph entry for the exact selected shared event name.

You do not also need a Run Visual Script call to make this subscription listen. The picker exposes 15 names; the text-script subscription panel has 18.

Field / stored key	Input	Default where specified
Event `eventName`	Choose: OnGameStart, OnSceneLoaded, OnSceneLeaving, OnPlayerChanged, OnCharacterControl, OnItemFound, OnDialogueStarted, OnDialogueEnded, OnInventoryChanged, OnSaveLoaded, OnCustomEvent, OnInteraction, OnTriggerEntered, OnTriggerExited, OnAttachPointContact.	-

IfCompare

Compare two resolved values with ==, !=, <, <=, > or >=.

For score >= 10, use {"$var":"score"}, >= and numeric 10. Text ordering is not numeric ordering.

Field / stored key	Input	Default where specified
Left `left`	A typed value or valid JSON.	{"$var": "score"}
Compare `operator`	Choose: ==, !=, <, <=, >, >=.	-
Right `right`	A typed value or valid JSON.	1

ReadContext

Copy one event-context value into a game variable.

Read reason into last_reason for a custom event. Missing keys can produce null; check the emitter's payload.

Field / stored key	Input	Default where specified
Context Key key	Text.	"entityId"
Store in Variable name	Text.	"target"

CallScript

Call another script/function with a JSON argument array and optionally store its result.

This is a graph-only bridge with different parameter fields from Functions as Nodes.

Field / stored key	Input	Default where specified
Script scriptId	Choose a visual or supported text script.	-
Function (optional) functionName	Text.	-
Arguments (JSON array) arguments	A typed value or valid JSON.	[]
Store Result in Variable resultVariable	Name of the game variable that receives the returned value.	-

RandomValue

Generate a random number in the selected range and store it.

Enable Whole Numbers for an integer-style choice. Keep the range small enough to test every outcome.

Field / stored key	Input	Default where specified
Minimum `minimum`	Number; decimals allowed.	0.0
Maximum `maximum`	Number; decimals allowed.	1.0
Whole Numbers `wholeNumbers`	Store an integer-style random result instead of a fractional result.	false
Store in Variable `resultVariable`	Name of the game variable that receives the returned value.	"random_value"

GetIncludedVariable

Read a shared variable from an included/shared script and optionally store it.

This is script-shared state, not simply another name for a project game variable.

Field / stored key	Input	Default where specified
Script `scriptId`	Choose a visual or supported text script.	-
Shared Variable `variableName`	Text.	-
Store in Variable `resultVariable`	Name of the game variable that receives the returned value.	-

SetIncludedVariable

Write a shared variable on an included/shared script.

Use the script ID and the exact shared variable name; choose a value of its intended type.

Field / stored key	Input	Default where specified
Script `scriptId`	Choose a visual or supported text script.	-
Shared Variable `variableName`	Text.	-
Value (JSON) `value`	A value of the type required by the selected variable, operation or property.	null

Return

Return a value from the visual graph.

Use it when a calling function or script needs a result. Returning false does not automatically cancel the rest of an unrelated action list.

Field / stored key	Input	Default where specified
Return Value `value`	A value of the type required by the selected variable, operation or property.	null

Wait

Wait a number of seconds before continuing graph flow.

Use a short Wait after starting an animation if you need a simple pause. For overlapping camera/audio/animation beats, a cutscene timeline is clearer.

Field / stored key	Input	Default where specified
Seconds `seconds`	Number; decimals allowed.	0.25

EmitEvent

Emit an exact event name with a JSON context object.

Choose OnCustomEvent with {"reason":"signal_powered"}. Broadcasting from its own matching listener without a guard can recurse.

Field / stored key	Input	Default where specified
Event Name `eventName`	Text.	"OnCustomEvent"
Event Context (JSON object) `context`	A typed value or valid JSON.	{}

How to read the action reference

There are 211 nonblank entries in the shared sequence picker, plus five additional timeline-specific entries. The reference covers the union: 216 commands. Availability is checked separately for the native inspector, shared list/GUI, visual graph, text wrapper and Cutscene Timeline.

Reference label	Meaning
Native inspector	Offered by the native room/object/trigger action picker. Individual surfaces can restrict a few entries further.
Shared list / GUI	Listed by the reusable sequence picker used for additional GUI actions.
Visual graph	Available as a visual action node; conditions and flow nodes have their own preceding reference.
Text action wrapper	Included in the common supported scripting action surface.
Timeline	Listed in the Cutscene Timeline's Add Event choices.

The field tables show the common catalog schema, supplemented by important specialized fields where a designer supplies them. Starting values describe the catalog's initial data, not a guarantee that every editor surface or existing project displays that same value. Set the values you want explicitly.

A blank reference field needs a real project ID where that command requires one. Selecting a scene to populate a picker does not necessarily make an action work on an unloaded room. Check the entry's use notes for actions that operate on current live entities.

> The reference describes the checked 0.86.3.4 build. Listed availability does not mean that every possible combination of properties, imported formats and renderer works identically on every device. The version notes identify specific editor limitations found while preparing these lessons.

Variables, rewards and reusable logic

Keep puzzle facts in variables, then use a visual or text script when a decision is needed. A list is an ordered set of instructions; it does not gain a branch just because an action returns false.

Set Variable

ID: `SetVariable` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Assign a value to a runtime game variable. This replaces its previous value and can create the runtime entry if it does not exist.

Field / stored key	What to enter	Starting value
Variable `name`	Choose a game variable name. Custom values are accepted.	Blank
Value `value`	A value of the type required by the selected variable, operation or property.	true

In use. Set powered to true when the lever is used. Use a Boolean value, not the text "true". In the room/object inspector, choose Change Variable with Operation set to Set instead.

Create Variable

ID: `CreateVariable` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Create a variable only when that name is absent. It is an initializer, so calling it again does not reset an existing value.

Field / stored key	What to enter	Starting value
Variable ID `name`	Text.	Blank
Type `variableType`	Choose: string, boolean, number, array.	"string"
Initial Value `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Create visits with a numeric value of 0, then increment it on room entry. Define long-lived puzzle variables in the Variable Designer so their starting values are easy to find. The variableType field is descriptive; do not rely on it to convert bad input.

Change Variable

ID: `ChangeVariable` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set or modify a variable using set, arithmetic, append or toggle. The operation determines how Value is used.

Field / stored key	What to enter	Starting value
Variable `name`	Choose a game variable name.	Blank
Operation `operation`	Choose: set, add, subtract, multiply, divide, append, toggle.	"set"
Value `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Choose score, Add, 5 for a five-point reward. Toggle suits a repeatable switch; Set is safer when an event may run twice. Keep arithmetic numeric and avoid dividing by zero.

Add Score

ID: `AddScore` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Increase the runtime variable named score by the given integer amount.

Field / stored key	What to enter	Starting value
Amount `amount`	Number of points to add to score.	1

In use. Award 5 after collecting a unique key. This action does not remember that the key was already collected. Put it behind the same one-time condition as the reward. In a native inspector, Change Variable / score / Add is the direct equivalent.

Unlock Hint Topic / Answer

ID: `UnlockHint` | Shared list / GUI · Visual graph · Text action wrapper

Unlock a numbered answer within an existing hint topic.

Field / stored key	What to enter	Starting value
Topic ID `topicId`	Text.	Blank
Answer Number `answerNumber`	Whole number.	1

In use. Unlock answer 2 for the archive topic after the player has examined the door. Match the topic ID and answer numbering used by the hint system; an invented ID does not create a complete hint topic.

Run Visual Script

ID: `RunVisualScript` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Run a selected script and wait for that call. Despite its label, the picker can select the supported text-script languages as well as visual graphs.

Field / stored key	What to enter	Starting value
Script `scriptId`	Choose a visual or supported text script.	Blank

Timing: Waits for the selected script call.

In use. Put the door's inventory check in open_door, then call that script from the door's Use response. Keep the conditional logic in that one place instead of copying it into several event lists.

Call Script Function

ID: `CallScriptFunction` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Call a function published through Functions as Nodes, pass named parameters, and optionally store its return value.

Field / stored key	What to enter	Starting value
Function `functionKey`	Text/reference.	Blank
Parameters `parameters`	A typed value or valid JSON.	`{}`
Result Variable `resultVariable`	Name of the game variable that receives the returned value.	Blank

Timing: Waits for the published function call.

In use. Publish a reward function with points and message parameters. Choose it here, enter those parameters, and store the result in reward_ok. The function must be defined and published first; this action does not infer a function from an arbitrary name.

End Visual Script

ID: `EndVisualScript` | Native inspector · Shared list / GUI · Visual graph

Request that the running visual graph finish. Also End Calling Script controls whether the parent visual call should end too.

Field / stored key	What to enter	Starting value
Also End Calling Script `endParent`	Also request termination of the calling visual graph.	true

In use. Use it on an early-out branch inside a graph. It is not a general break statement for a room or GUI action array: the ordinary array runner continues iterating its remaining entries. Use graph End or a script return for a clear branch exit.

Inventory and two-object interactions

An inventory definition describes an item. Giving an item changes a bag. Removing its scene model is a separate instruction. Decide whether the project uses shared or character-specific inventory before writing transfers.

Add Inventory

ID: `AddInventory` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Add copies of an existing inventory item through the runtime's active inventory route.

Field / stored key	What to enter	Starting value
Item `itemId`	Choose an inventory definition ID.	Blank
Count `count`	Number of copies or repetitions. Use a whole number.	1

In use. Add one Archive Key after a successful pickup. This is not a command to import a model, and it does not hide the key on the workbench. Use Give Inventory in the room/object inspector when this entry is not listed there.

Give Inventory

ID: `GiveInventory` | Native inspector · Shared list / GUI

Give a quantity of an item to the selected character's inventory route.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Item `itemId`	Choose an inventory definition ID.	Blank
Count `count`	Number of copies or repetitions. Use a whole number.	1

In use. Choose Player, Archive Key and Quantity 1 for the pickup lesson. In character-owned inventory games, choose the intended character explicitly. A shared-inventory setting can make character ownership appear unchanged because everyone is using the same bag.

Remove Inventory

ID: `RemoveInventory` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Remove copies of an item through the current inventory route.

Field / stored key	What to enter	Starting value
Item `itemId`	Choose an inventory definition ID.	Blank
Count `count`	Number of copies or repetitions. Use a whole number.	1

In use. Remove one fuse after it has successfully been installed. Test the condition first, then remove it. Removing a missing item is not a substitute for a Has Inventory branch, and the next action in an ordinary list will still run.

Change Inventory Quantity

ID: `ChangeInventoryQuantity` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Apply a signed quantity change to an item for a character.

Field / stored key	What to enter	Starting value
Player / Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Inventory Item `itemId`	Choose an inventory definition ID.	Blank
Change By `amount`	Signed quantity change: positive adds, negative removes.	1

In use. Use amount -1 to consume a battery, or +3 to grant three coins. This is a change by an amount, not an instruction to set an absolute total. Choose the same owner that the inventory GUI is displaying.

Change Inventory from Player to Player

ID: `TransferInventory` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Move a quantity from one character's inventory to another.

Field / stored key	What to enter	Starting value
From Player `fromCharacterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
To Player `toCharacterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Inventory Item `itemId`	Choose an inventory definition ID.	Blank
Quantity `count`	Number of copies or repetitions. Use a whole number.	1

In use. Transfer the key from Ivo to Elias in a game with individual bags. Check the source character and quantity before playing a handover animation. Changing which character is controlled is not the same thing as transferring an item.

Execute Two-Object Command

ID: `ExecutePreposition` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Perform a two-target verb interaction using a source, target and verb.

Field / stored key	What to enter	Starting value
Verb `verbId`	Choose a Verb Designer ID.	Blank
First Object (entity:id or inventory:id) `sourceId`	Text.	Blank
Second Object (entity:id or inventory:id) `targetId`	Text.	Blank

In use. For Use Key On Door, supply the source item/record reference, the door reference, and the Use verb. Configure the verb's preposition rules and the matching interaction first. A normal scene entity ID and an inventory reference are different kinds of target.

Cancel Two-Object Command

ID: `CancelPreposition` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Cancel the currently assembled two-object interaction.

No fields in the standard catalog form.

In use. Let a Cancel button clear Use Key On ... when the player changes their mind. It clears the pending combination, rather than deleting the item or disabling the verb permanently.

Execute Selected Verb Command

ID: `ExecuteVerb` | Native inspector · Shared list / GUI

Run the currently selected verb against an entity. The native label is Execute Selected Verb Command.

Field / stored key	What to enter	Starting value
Target (structured calls only) `entityId`	Choose a scene entity ID.	-

In use. Use it in a trigger's click response to hand control back to that trigger's normal Verb Actions. The native event supplies the entity. A standalone structured call can supply entityId, but the standard catalog shows no separate target fields. Avoid calling it recursively from the same verb's own response.

Speech, messages, voice and dialogue

Use a message for narration, a speech action for a speaking character, and Start Dialogue for a conversation graph. These are different routes to text. Typewriter controls are shared by the message and speech actions and by newly added dialogue lines and choices.

Show Message

ID: ShowMessage | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Display narration or a short message for a duration, optionally blocking player input.

Field / stored key	What to enter	Starting value
Text `text`	Text.	Blank
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits.	1.8
Block Player Input `blockInput`	Block normal player input while the message is presented.	false
Voice Audio `voiceAssetId`	Choose an imported audio asset. May be blank.	Blank
Voice Key (optional) `voiceKey`	Optional localization/voice lookup key.	Blank
Typewriter Effect `typewriter`	Reveal text progressively instead of displaying it all at once.	false
Typing Speed (characters/second) `typewriterSpeed`	Text reveal rate in characters per second. Range: 1.0 to 240.0.	30.0
Typing Sound `typewriterSound`	Choose: default, none, custom.	"default"
Typing Sound Asset `typewriterSoundAssetId`	The imported audio used when the typing sound is custom. May be blank.	Blank

Typewriter: speed is characters per second; sound can be default, none or custom. Custom uses the selected typing sound asset.

Timing: Waits for the message routine.

In use. Show "The signal is on." for 2.2 seconds. Block Player Input affects movement/interaction while the message is up; the action call itself still waits for the message routine. Keep long conversations in the Dialogue Designer.

Say

ID: Say | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Speak as the chosen character. With Wait for Continue off, speech uses timed advancement; with it on, the player must continue.

Field / stored key	What to enter	Starting value
Speaker `speakerId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Text `text`	Text.	Blank
Wait for Continue `waitForContinue`	Wait for the player to continue the speech instead of using timed advancement.	false
Voice Audio `voiceAssetId`	Choose an imported audio asset. May be blank.	Blank
Voice Key (optional) `voiceKey`	Optional localization/voice lookup key.	Blank
Typewriter Effect `typewriter`	Reveal text progressively instead of displaying it all at once.	false
Typing Speed (characters/second) `typewriterSpeed`	Text reveal rate in characters per second. Range: 1.0 to 240.0.	30.0
Typing Sound `typewriterSound`	Choose: default, none, custom.	"default"
Typing Sound Asset `typewriterSoundAssetId`	The imported audio used when the typing sound is custom. May be blank.	Blank

Typewriter: speed is characters per second; sound can be default, none or custom. Custom uses the selected typing sound asset.

Timing: Waits for the speech routine; manual continuation is optional.

In use. Let Elias explain the machine, then run the next instruction. A timed line without a custom duration uses the runtime's short speech timing. Voice and typewriter settings affect presentation; the speaker still needs a valid scene or character definition.

Character Say

ID: `CharacterSay` | Native inspector · Shared list / GUI

Speak as a particular character, using that character's speech presentation and lip-sync setup.

Field / stored key	What to enter	Starting value
Speaker `speakerId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Text `text`	Text.	Blank
Wait for Continue `waitForContinue`	Wait for the player to continue the speech instead of using timed advancement.	false
Voice Audio `voiceAssetId`	Choose an imported audio asset. May be blank.	Blank
Voice Key (optional) `voiceKey`	Optional localization/voice lookup key.	Blank
Typewriter Effect `typewriter`	Reveal text progressively instead of displaying it all at once.	false
Typing Speed (characters/second) `typewriterSpeed`	Text reveal rate in characters per second. Range: 1.0 to 240.0.	30.0
Typing Sound `typewriterSound`	Choose: default, none, custom.	"default"
Typing Sound Asset `typewriterSoundAssetId`	The imported audio used when the typing sound is custom. May be blank.	Blank

Typewriter: speed is characters per second; sound can be default, none or custom. Custom uses the selected typing sound asset.

Timing: Waits for the speech routine; manual continuation is optional.

In use. Choose Elias and enter one short observation. Enable Wait for Continue only when the player has a usable Continue control. In native object inspectors this action also exposes a duration field for the timed form.

Player Say

ID: `PlayerSay` | Native inspector · Shared list / GUI

Have the currently controlled character speak the supplied text.

Field / stored key	What to enter	Starting value
Text `text`	Text.	Blank
Voice Audio `voiceAssetId`	Choose an imported audio asset. May be blank.	Blank
Voice Key (optional) `voiceKey`	Optional localization/voice lookup key.	Blank
Typewriter Effect `typewriter`	Reveal text progressively instead of displaying it all at once.	false
Typing Speed (characters/second) `typewriterSpeed`	Text reveal rate in characters per second. Range: 1.0 to 240.0.	30.0
Typing Sound `typewriterSound`	Choose: default, none, custom.	"default"
Typing Sound Asset `typewriterSoundAssetId`	The imported audio used when the typing sound is custom. May be blank.	Blank

Typewriter: speed is characters per second; sound can be default, none or custom. Custom uses the selected typing sound asset.

Timing: Waits for timed player speech.

In use. Use "I need the archive key." when a door is locked. It follows the active player rather than a hard-coded protagonist, which is useful in a game with character switching. This action's normal catalog form is timed, not a click-to-continue dialogue node.

Set Voice Speaker

ID: `SetVoiceSpeaker` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Choose the default character whose mouth follows separately played voice audio or message voice.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported. May be blank.	Blank

In use. Select Elias before Play Voice. Clear the selection when that default is no longer wanted. Say and Character Say already name their own speaker; configure the character's mouth and portrait mapping in the Character Designer first.

Enable Lip Sync

ID: `SetLipSyncEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Enable or disable lip sync for a character at runtime.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable lip sync for a silent disguise sequence, then enable it afterward. This is a runtime switch, not a replacement for defining visemes, portrait mouth frames or blend shapes. Turning it off restores the resting face.

Play Voice

ID: `PlayVoice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Play voice audio selected by asset or localization key, optionally waiting for it to finish.

Field / stored key	What to enter	Starting value
Voice Audio `voiceAssetId`	Choose an imported audio asset. May be blank.	Blank
Voice Key (optional) `voiceKey`	Optional localization/voice lookup key.	Blank
Text `text`	Text.	Blank
Wait for Voice `wait`	Whether this action should wait for its own effect to finish.	false

Timing: Waits when Wait for Voice is on.

In use. Play a radio line while the scene remains visible. If the next instruction must follow the recording, enable Wait for Voice. Text helps the built-in text/voice mouth workflow; it does not automatically turn this into a full dialogue conversation.

Pause Voice

ID: `PauseVoice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Pause the current voice and its mouth timing.

No fields in the standard catalog form.

In use. Pause a recording while a message is being inspected. A conversation awaiting that recording remains in progress; use Stop Voice if the recording is meant to be cancelled instead.

Resume Voice

ID: `ResumeVoice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Resume paused voice playback and the shared scene/portrait mouth timing.

No fields in the standard catalog form.

In use. Pair it with Pause Voice. It resumes the current recording rather than choosing a new asset. Calling it without a paused voice does not start a new line.

Seek Voice

ID: `SeekVoice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Move the current voice playback position to a time in seconds.

Field / stored key	What to enter	Starting value
Seconds `seconds`	Number; decimals allowed.	0.0

In use. Seek to 3.0 seconds while checking a line's mouth cues. Test the associated text and timing track at the new position, especially when a conversation is also waiting for audio completion.

Stop Voice

ID: `StopVoice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Stop the current voice and reset its mouth presentation.

No fields in the standard catalog form.

In use. Use it when an interrupted radio message should end immediately. It is voice-specific; sound effects and music use their own stop actions.

Start Dialogue

ID: `StartDialogue` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Start a Dialogue Designer conversation and wait for that dialogue session to finish.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID.	Blank

Timing: Waits for the dialogue session.

In use. On Elias's Talk verb, choose Ask Elias. The dialogue record decides its start node, choices and flow. Do not start the same dialogue again in OnDialogueStarted, which can create nested or repeated conversations.

End Dialogue

ID: `EndDialogue` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

End the active dialogue session.

No fields in the standard catalog form.

In use. Use it when a scripted event cuts a conversation short. It does not erase the authored dialogue, its choices or the saved read history. Test any actions that were supposed to follow a normal ending.

Add Dialogue Line

ID: `AddDialogueLine` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Insert or update a line node in a runtime dialogue graph, including speaker, text, next node, voice and typewriter options.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID.	Blank
New Node ID `nodeId`	Text.	Blank
Speaker `speakerId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Text `text`	Text.	Blank
Next Node `next`	Choose a node in that dialogue. May be blank. Custom values are accepted.	Blank
Return to previous node `returnToPrevious`	Return through dialogue history rather than relying only on a fixed next link.	true
Voice Audio `voiceAssetId`	Choose an imported audio asset. May be blank.	Blank
Voice Key (optional) `voiceKey`	Optional localization/voice lookup key.	Blank
Typewriter Effect `typewriter`	Reveal text progressively instead of displaying it all at once.	false
Typing Speed (characters/second) `typewriterSpeed`	Text reveal rate in characters per second. Range: 1.0 to 240.0.	30.0
Typing Sound `typewriterSound`	Choose: default, none, custom.	"default"

Field / stored key	What to enter	Starting value
Typing Sound Asset `typewriterSoundAssetId`	The imported audio used when the typing sound is custom. May be blank.	Blank

Typewriter: speed is characters per second; sound can be default, none or custom. Custom uses the selected typing sound asset.

In use. Add an answer node named signal_answer, then point a choice at it. Use a unique stable node ID. Return to previous node is enabled by default, so check that behavior before also relying on a fixed Next link.

Add Dialogue Choice Node

ID: `AddDialogueChoiceNode` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Create a choice node in an existing dialogue.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID.	Blank
New Node ID `nodeId`	Text.	Blank

In use. Create extra_topics, then add choices to that node with Add Dialogue Choice. Creating the choice node alone does not make the player visit it: connect an existing node to it or otherwise change the graph flow.

Remove Dialogue Node

ID: `RemoveDialogueNode` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Remove a node from the runtime dialogue graph.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID.	Blank
Dialogue Node ID `nodeId`	Choose a node in that dialogue. Custom values are accepted.	Blank

In use. Remove a temporary topic after its story purpose is finished. Check every Next link and choice that points to it; removing a destination can leave a conversation with nowhere useful to go.

Add Dialogue Choice

ID: `AddDialogueChoice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Add or update an option inside a choice node.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID.	Blank
Choice Node ID `nodeId`	Choose a choice node in that dialogue. Custom values are accepted.	Blank
New Choice ID `choiceId`	Text.	Blank
Text `text`	Text.	Blank
Next Node `next`	Choose a node in that dialogue. May be blank. Custom values are accepted.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true
Always Allow Repeat `repeatable`	Allow this dialogue choice to be selected repeatedly under its read policy.	false
Voice Audio `voiceAssetId`	Choose an imported audio asset. May be blank.	Blank
Voice Key (optional) `voiceKey`	Optional localization/voice lookup key.	Blank
Typewriter Effect `typewriter`	Reveal text progressively instead of displaying it all at once.	false
Typing Speed (characters/second) `typewriterSpeed`	Text reveal rate in characters per second. Range: 1.0 to 240.0.	30.0
Typing Sound `typewriterSound`	Choose: default, none, custom.	"default"
Typing Sound Asset `typewriterSoundAssetId`	The imported audio used when the typing sound is custom. May be blank.	Blank

Typewriter: speed is characters per second; sound can be default, none or custom. Custom uses the selected typing sound asset.

In use. Add a "The signal is working" option after powered becomes true. Choose the dialogue, choice node, new choice ID and next node. Repeatable controls repeated availability; it is separate from whether the choice is currently enabled.

Remove Dialogue Choice

ID: `RemoveDialogueChoice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Remove one option identified by its choice ID from a choice node.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID.	Blank
Choice Node ID `nodeId`	Choose a choice node in that dialogue. Custom values are accepted.	Blank
Choice ID `choiceId`	Choose an option ID in that choice node. Custom values are accepted.	Blank

In use. Remove an introductory question once it no longer makes sense. For a choice that may return later, Set Dialogue Choice Enabled is usually easier to reverse than deleting and recreating it.

Enable / Disable Dialogue Choice

ID: `SetDialogueChoiceEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Enable or disable an existing choice without removing its record.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID.	Blank
Choice Node ID `nodeId`	Choose a choice node in that dialogue. Custom values are accepted.	Blank
Choice ID `choiceId`	Choose an option ID in that choice node. Custom values are accepted.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Enable the archive question after examining the door. Select the exact choice node and choice ID. Its visible/disabled presentation also depends on the dialogue read and choice-display settings.

Reset Dialogue History

ID: `ResetDialogueHistory` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Clear remembered read/use history for a dialogue, node or choice scope.

Field / stored key	What to enter	Starting value
Dialogue `dialogueId`	Choose a Dialogue Designer ID. May be blank.	Blank
Dialogue Node ID `nodeId`	Choose a node in that dialogue. May be blank. Custom values are accepted.	Blank
Choice ID `choiceId`	Choose an option ID in that choice node. May be blank. Custom values are accepted.	Blank

In use. Reset one repeatable hint conversation during a new puzzle phase. Blank scopes can broaden the reset; select the narrowest scope you need so unrelated conversations do not suddenly become unread.

Characters, movement and animation

Movement commands act on characters that the runtime can resolve. A destination can be a target record or coordinates. The room picker helps resolve that destination; it is not a promise that the character will walk through unloaded rooms.

Character Walk To

ID: `CharacterWalkTo` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Walk a character to the chosen target or position, waiting for the movement call.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Object Room `sceneId`	Choose a scene ID.	Blank
Object (optional) `targetId`	Choose an entity in the selected scene. May be blank.	Blank
XYZ `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]
Walking Animation `animation`	Choose a clip/state available to that entity. May be blank.	Blank
Speed `speed`	Movement speed; 0 uses the normal movement speed where supported.	0

Timing: Waits for the movement routine.

In use. Walk Elias to an approach marker beside the workbench. Use a reachable marker instead of the middle of a solid door. The current action bridge uses the character's movement logic; the Animation field is not a reliable override for its walk-state setup.

Character Waypoints: Clear Queue

ID: `ClearCharacterWaypoints` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Empty a character's queued destinations.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank

In use. Clear an NPC's old patrol before adding a new route. This edits the queue, rather than providing a visible destination or starting a new walk by itself.

Character Waypoints: Add Waypoint

ID: `AddCharacterWaypoint` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Append one destination to a character's waypoint queue.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Waypoint Room `sceneId`	Choose a scene ID.	Blank
Waypoint / Object (optional) `waypointId`	Choose an entity in the selected scene. May be blank.	Blank
XYZ when no waypoint `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]
Ignore Navigation Mesh `ignoreNavigation`	Use the direct route instead of following normal navigation constraints.	false

In use. Add three named markers around the room, then call Character Walk Waypoints. Ignore Navigation permits a direct route; keep it off for ordinary floor walking so obstacles and walkable areas remain meaningful.

Character Waypoints: Walk Queue

ID: `CharacterWalkWaypoints` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Run the queued waypoint route for a character.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Speed `speed`	Movement speed; 0 uses the normal movement speed where supported.	0

Timing: Runs the queued movement route.

In use. After clearing and adding the patrol markers, choose Elias and start the queue. Building the queue and walking it are separate steps. A blocked marker can prevent the intended route from completing.

Character Walk Through Waypoints

ID: `CharacterWalkPath` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Walk through an array of waypoint references, optionally ignoring navigation.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Default Waypoint Room `sceneId`	Choose a scene ID.	Blank
Waypoint IDs / Positions `waypointIds`	JSON array of waypoint references in travel order.	[]
Speed `speed`	Movement speed; 0 uses the normal movement speed where supported.	0
Ignore Navigation Mesh `ignoreNavigation`	Use the direct route instead of following normal navigation constraints.	false

Timing: Runs the selected movement route.

In use. Supply a JSON array of marker IDs in route order, then choose the walking speed. Use local, reachable markers for a first test. A list of display labels is not a list of the stored IDs.

Character Instant Teleport To

ID: `CharacterTeleport` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Move a character directly to a target or coordinates without walking there.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Object Room `sceneId`	Choose a scene ID.	Blank
Object (optional) `targetId`	Choose an entity in the selected scene. May be blank.	Blank
XYZ `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]

In use. Place the player at a staged position before a scene reveal. Check the floor height and collider clearance; teleporting into a wall is still possible. Use Change Scene for a player room transition and Move NPC To Room for an NPC's room assignment.

Move NPC to a Different Room

ID: `MoveNPCToRoom` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change an NPC's room assignment and arrival position or spawn.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Room `sceneId`	Choose a scene ID.	Blank
Spawn (optional) `spawnId`	Choose an arrival spawn in the destination scene. May be blank.	Blank
XYZ when no spawn `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]

In use. Send Elias to the archive after the puzzle. This is a world-state change rather than visible walking between rooms. Confirm that the destination has the intended spawn and that a scene does not already contain a conflicting copy of the NPC.

Follow Character

ID: `FollowCharacter` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Start one character following another with a chosen spacing and speed.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Character to Follow `targetCharacterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Distance Behind `distance`	Number; decimals allowed.	0
Speed `speed`	Movement speed; 0 uses the normal movement speed where supported.	0

In use. Have Elias follow the player at about 1.5 units. It starts a continuing behavior; the next list instruction need not wait for a final destination. Stop the follower before a staged conversation if both actors must hold their marks.

Stop Follow Character

ID: `StopFollowCharacter` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Stop the chosen character's follow behavior.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank

In use. Call it before walking an NPC to a specific scene marker. It stops the following relationship; it does not remove the character or disable future movement commands.

Face Character

ID: `FaceCharacter` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Turn one character toward another.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Face Character `targetCharacterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank

In use. Face Elias toward the player before his greeting. Select the acting character first and the target second. Turning alone does not bring them closer or guarantee a clear line of sight.

Face Object

ID: `FaceObject` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Turn a character toward a scene object.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Object Room `sceneId`	Choose a scene ID.	Blank
Object `targetId`	Choose an entity in the selected scene.	Blank

In use. Make the player face the signal lamp after using the lever. The target's position determines the facing direction; a large object's origin may not be the point you intended.

Face Coordinates

ID: `FaceCoordinates` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Turn a character toward an explicit world position.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
XYZ `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]

In use. Use a position near the door handle when the door model's origin is unhelpful. In a 3D scene, provide X, Y and Z. This is a facing target, not a rotation triplet.

Hide Player Character

ID: `HidePlayerCharacter` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Hide or reveal the active player's visual representation.

Field / stored key	What to enter	Starting value
Hidden `hidden`	Whether the quest or player visual should be hidden for this action.	true

In use. Hide the player during a first-person inspection and reveal them on exit. Hiding the model is not an input lock and does not, by itself, stop the player moving.

Enable / Disable Player Input

ID: `SetPlayerInputEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable or disable player control input.

Field / stored key	What to enter	Starting value
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable control during a carefully staged action and restore it on every exit path. Other locks, such as a running dialogue or cutscene, can still keep input unavailable after this action enables its own switch.

Enable / Disable Character Action

ID: `SetCharacterActionEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable or disable a named interaction response on a character.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Mouse / Trigger Action `actionEvent`	Choose the response/event to enable or disable.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable Talk for a character who is busy, then restore it later. The event/verb selection matters: disabling one response is different from hiding the NPC or turning off all interaction.

Change Character Property

ID: `ChangeCharacterProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a selected character property in runtime data.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Property `property`	Choose a supported character property.	Blank
To `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Change a character's display or movement property supported by the picker. Use a value of the right type. A generic property assignment is not guaranteed to rebuild an already loaded model, material or animation setup.

Add Character To Scene

ID: `AddCharacterToScene` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Create a scene instance from an existing Character Designer definition.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
New ID `id`	Text.	Blank
Name `name`	Text.	Blank
XYZ `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]
Enabled `enabled`	Explicit on/off state for the feature or response.	true
Visible `visible`	Explicit visible/hidden state.	true
Animation (optional) `animation`	Text.	Blank

In use. Add Elias at [2,0,-1] with a unique scene entity ID. Choose the character definition as the source; the scene instance's ID identifies the copy you create. Repeating this action without an existence check can create unwanted duplicates.

Jump

ID: `Jump` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Ask the selected character to jump through the adventure extension.

Field / stored key	What to enter	Starting value
Character `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank

In use. Use it for an appropriate controllable character during a test. A jump action does not automatically provide a jump animation, a platforming controller or a valid landing surface; those still depend on the character and room setup.

Play Animation

ID: `PlayAnimation` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Request an animation clip or state on the target entity. Most imported clips start and return immediately.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Character / Object `entityId`	Choose an entity in the selected scene.	Blank
Animation `clip`	Choose a clip/state available to that entity.	Blank

Timing: Most imported animation requests return after starting playback. They do not wait for the full clip length; some procedural fallback motions do wait.

In use. Use a short, non-looping lever gesture or nod. Confirm the clip/state ID exists on the actual imported model. Looping animations do not have a natural one-shot finish, so do not build story timing on an endless loop.

Stop Animation

ID: `StopAnimation` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Stop the target entity's current animation through the runtime animation route.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Character / Object `entityId`	Choose an entity in the selected scene.	Blank

In use. Stop a looping inspection motion before changing the scene. For normal idle/walk behavior, allow the character's state system to resume afterward rather than assuming the stopped pose is a permanent character state.

Animation: Pause

ID: `PauseAnimation` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Pause the target entity's animation playback.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Character / Object `entityId`	Choose an entity in the selected scene.	Blank

In use. Freeze an inspection model while the player reads a note. Pausing a clip does not pause the whole game, music or the character's navigation.

Animation: Resume

ID: `ResumeAnimation` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Resume the target entity's paused animation playback.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Character / Object `entityId`	Choose an entity in the selected scene.	Blank

In use. Pair it with Pause Animation on the same entity. If another state or clip replaced the paused animation, reselecting the correct state may be necessary.

Objects, descriptions and scene life

Visible, enabled, interactive and physically blocking are separate ideas. Use the smallest change that matches the story. For an item pickup, hiding its world representation and giving its inventory definition are two distinct actions.

Enable / Disable Object Action

ID: `SetObjectActionEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable or disable one object's interaction event or verb response.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Object `entityId`	Choose an entity in the selected scene.	Blank
Mouse / Trigger Action `actionEvent`	Choose the response/event to enable or disable.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable Use after a machine has been repaired while keeping Look available. Choose the intended event. This is more precise than disabling the entire object.

Enable / Disable Object

ID: `SetObjectEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set an object's enabled state.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Object `entityId`	Choose an entity in the selected scene.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable a puzzle object while the scene is being prepared. Test its visual, input and collision behavior in the running game; use Set Object Visible when you only mean to change whether it is drawn.

Hide / Unhide Object

ID: `SetObjectVisible` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Show or hide an object.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Object `entityId`	Choose an entity in the selected scene.	Blank
Visible `visible`	Explicit visible/hidden state.	true

In use. Hide the workbench key after giving its inventory item. Hiding is not a collision-size editor or a substitute for disabling a trigger. For reusable pickup logic, target this in a script with an object caller.

Enable / Disable Trigger

ID: `SetTriggerEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable or disable a trigger's runtime event behavior.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Trigger `entityId`	Choose a trigger in the selected scene.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Turn off a doorway trigger during an opening cutscene, then turn it on when play begins. Existing overlaps and the player starting inside the volume need to be tested; do not treat the toggle as a guaranteed fresh Enter event.

Spawn Scene Object

ID: `SpawnObject` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Spawn or restore an already authored scene object definition.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Scene Object `entityId`	Choose an entity in the selected scene.	Blank

In use. Restore a temporary prop that was despawned. The chosen object must have an authored definition the runtime can find. Use Create Primitive or Add Asset To Scene to create a new record instead.

Despawn Scene Object

ID: `DespawnObject` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Remove a live object instance through the runtime scene-life route.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Scene Object `entityId`	Choose an entity in the selected scene.	Blank

In use. Remove a temporary projection when its sequence ends. Keep an ID or authored definition if it will be restored. Despawning a visible item does not automatically remove a matching inventory item.

Change Object Property

ID: `ChangeObjectProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Assign a property on an object's runtime record.

Field / stored key	What to enter	Starting value
Scene `sceneId`	Choose a scene ID.	Blank
Object `entityId`	Choose an entity in the selected scene.	Blank
Property `property`	Choose a supported object property.	Blank
To `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Change a supported property such as a display field or simple state value. Dedicated visibility, animation and effect actions are preferable when they exist: a generic record edit may not rebuild the corresponding live visual resource.

Change Scene Description

ID: `ChangeSceneDescription` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change the description stored for a scene.

Field / stored key	What to enter	Starting value
Scene `sceneId`	Choose a scene ID.	Blank
Description `description`	Text.	Blank

In use. Replace the room description after the machine starts. This edits descriptive text; it is not an automatic speech line. Add Show Message or Player Say if the player should hear or read it immediately.

Change Object Description

ID: `ChangeObjectDescription` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change an object's descriptive text.

Field / stored key	What to enter	Starting value
Scene `sceneId`	Choose a scene ID.	Blank
Object `entityId`	Choose an entity in the selected scene.	Blank
Description `description`	Text.	Blank

In use. After powering the lamp, change its description to "The signal is powered." The next Look response must use that description or the built-in fallback; a hard-coded Look message continues to say whatever you wrote in it.

Change Inventory Description

ID: `ChangeInventoryDescription` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change an inventory item's description.

Field / stored key	What to enter	Starting value
Inventory Item `itemId`	Choose an inventory definition ID.	Blank
Description `description`	Text.	Blank

In use. Update the key's description after discovering which door it opens. Changing description text is separate from changing its icon, name or quantity.

Change Character Description

ID: `ChangeCharacterDescription` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a character's description.

Field / stored key	What to enter	Starting value
Character / Player `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank
Description `description`	Text.	Blank

In use. Describe Elias differently after he becomes a companion. A custom Look script with its own fixed text will not automatically use the changed description.

Change Region Description

ID: `ChangeRegionDescription` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change the description of a named region.

Field / stored key	What to enter	Starting value
Scene `sceneId`	Choose a scene ID.	Blank
Region `regionId`	Choose a region in the selected scene.	Blank
Description `description`	Text.	Blank

In use. Change a frozen pool's description after it melts. Choose the region in the correct scene. This does not turn a non-interactive region into a clickable hotspot.

Add Asset to Scene

ID: `AddAssetToScene` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Create a runtime scene entity using an imported asset.

Field / stored key	What to enter	Starting value
Asset `assetId`	Choose an imported asset.	Blank
New ID `id`	Text.	Blank
Name `name`	Text.	Blank
XYZ `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]
Enabled `enabled`	Explicit on/off state for the feature or response.	true
Visible `visible`	Explicit visible/hidden state.	true

In use. Spawn a lantern model at [1,0,-2] with a unique ID. Import and verify the asset first. Repeating the spawn is not the same as showing one existing lantern.

Create Primitive

ID: `CreatePrimitive` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Create a runtime primitive such as a box, sphere, cylinder, capsule or plane.

Field / stored key	What to enter	Starting value
Primitive `primitive`	Choose: box, sphere, cylinder, capsule, plane.	"box"
ID `id`	Text.	Blank
XYZ `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]

In use. Spawn a temporary marker for a puzzle demonstration. The normal action form has a small set of fields; detailed dimensions/material settings may need structured data or later property changes supported by the runtime.

Create Procedural

ID: `CreateProcedural` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Create a runtime procedural entity from the requested generator type.

Field / stored key	What to enter	Starting value
Procedural Type `procedural`	Text.	Blank
ID `id`	Text.	Blank
XYZ `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]

In use. Use a generator name supported by the engine and check the result in Test Scene. This is not a text-to-model system: an arbitrary description is not a procedural generator ID.

Cameras, transitions and screen effects

An authored camera is a reusable room view. A blend moves the view between cameras. An orbit or shake adds continuing or timed motion. Screen fades and transitions work on the presented image and need not change rooms.

Activate / Blend to Camera

ID: `ActivateCamera` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Activate an authored camera and optionally blend to it, waiting when Wait is enabled.

Field / stored key	What to enter	Starting value
Camera `cameraId`	Choose an authored camera ID.	Blank
Blend Seconds (-1 = camera default) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: -1.0 to 60.0.	-1.0
Wait for Blend `wait`	Whether this action should wait for its own effect to finish.	true

Timing: Waits when Wait is on.

In use. Blend to the lamp close-up over 1 second. A negative duration uses the camera's configured blend timing; 0 makes a cut. If a following action must begin after arrival, keep Wait enabled.

Set Advanced Camera Behavior

ID: `SetCameraBehavior` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Change the selected camera's advanced behavior, targets, smoothing, framing, path and bounds settings.

Field / stored key	What to enter	Starting value
Camera (empty = active) `cameraId`	Choose an authored camera ID. May be blank.	Blank
Enable Advanced Camera `enabled`	Explicit on/off state for the feature or response.	false
Camera Behavior `mode`	Choose: fixed, follow, group, framed, path.	"fixed"
Follow Target `targetId`	Choose a scene entity ID. May be blank.	"player"

Field / stored key	What to enter	Starting value
Additional Target IDs (comma separated) `targetIds`	Text.	Blank
Look At Target (optional) `lookAtId`	Choose a scene entity ID. May be blank.	Blank
Target Offset XYZ `targetOffset`	Offset from the camera target as [X, Y, Z].	[0.0, 1.3, 0.0]
Follow Damping (seconds) `damping`	Camera follow smoothing value; tune while watching the target move. Range: 0.0 to 5.0.	0.2
Default Camera Blend (seconds) `blendDuration`	Default camera blend time in seconds. Range: 0.0 to 60.0.	1.0
Blend Curve `ease`	Choose: linear, sine, cubic, quint.	"sine"
Camera Priority `priority`	Camera selection priority number. Range: -100000 to 100000.	0
Automatically Frame Group `autoFrame`	Allow the camera to calculate framing for its targets.	true
Minimum Group Distance / View Height `minDistance`	Nearest allowed camera framing distance. Range: 0.1 to 10000.0.	3.0
Maximum Group Distance / View Height `maxDistance`	Farthest allowed camera framing distance. Range: 0.1 to 10000.0.	18.0
Group Framing Padding `framingPadding`	Extra space around the framed target group. Range: 1.0 to 4.0.	1.3
Framing Zone Width (0-1) `deadZoneWidth`	Width of the camera framing/dead zone. Range: 0.0 to 1.0.	0.4
Framing Zone Height (0-1) `deadZoneHeight`	Height of the camera framing/dead zone. Range: 0.0 to 1.0.	0.3
Camera Path Points (XYZ array) `pathPoints`	Camera path point data as a JSON array; use the path editor when offered.	[]
Respect Room Camera Bounds `respectBounds`	Apply the room camera bounds to this camera behavior.	true

In use. Enable advanced behavior and choose Follow for a tracking shot. Changing the settings is different from activating that camera. Path mode needs usable path points; a group view needs valid targets. Keep Minimum Distance no greater than Maximum Distance.

Set Camera Targets

ID: `SetCameraTargets` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Replace the main, group and look-at targets used by an advanced camera.

Field / stored key	What to enter	Starting value
Camera (empty = active) `cameraId`	Choose an authored camera ID. May be blank.	Blank
Follow Target `targetId`	Choose a scene entity ID. May be blank.	"player"
Additional Target IDs (comma separated) `targetIds`	Text.	Blank
Look At Target (optional) `lookAtId`	Choose a scene entity ID. May be blank.	Blank

In use. Use the player as the follow target and Elias as the look-at target for a conversation. Supply IDs in the format the selected field expects. This does not create the targets or move either actor.

Set Camera Priority

ID: `SetCameraPriority` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Change the priority used by the advanced-camera system.

Field / stored key	What to enter	Starting value
Camera `cameraId`	Choose an authored camera ID.	Blank
Camera Priority `priority`	Camera selection priority number.	10

In use. Raise a conversation camera's priority while a scene is staged, then restore the previous priority. Priority is camera-selection policy; use Activate Camera when the sequence must explicitly take a particular view.

Change Camera

ID: `ChangeCamera` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Switch to a selected authored camera through the direct camera-change route.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Camera View `cameraId`	Choose a camera in the selected scene.	Blank

In use. Choose Room View after an inspection ends. The scene field supplies picker context; it does not transport the player to another room. Use Activate Camera when you want explicit blend duration and wait control.

Camera Orbit

ID: `CameraOrbit` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Start an orbit around an entity with distance, height, direction, speed and repetitions.

Field / stored key	What to enter	Starting value
Target Object `entityId`	Choose a scene entity ID.	Blank
Degrees Per Second `speed`	Orbit angular speed.	45
Distance `distance`	Number; decimals allowed.	5
Orbit Mode `mode`	Choose: 360, 180, 180_back.	"360"
Repeat Count `repetitions`	Whole number.	1
Clockwise `clockwise`	On/off.	false
Height `height`	Number; decimals allowed.	2

In use. Orbit the clock for a reveal, then stop it before returning to normal play. The orbit continues as a behavior after it starts; do not assume that placing it before another action makes the list wait for every revolution.

Stop Camera Orbit

ID: `StopCameraOrbit` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Stop the current extension-driven orbit.

No fields in the standard catalog form.

In use. Use it when an inspection closes or a conversation ends. Decide which normal camera should take over and activate that camera explicitly when needed.

Camera Shake

ID: `CameraShake` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Shake the 3D camera for the chosen duration and direction, waiting for the shake routine.

Field / stored key	What to enter	Starting value
Direction `direction`	Choose: both, horizontal, vertical.	"both"
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0.01 to 60.0.	0.5
Power `power`	Number; decimals allowed. Range: 0.0 to 10.0.	0.25

Timing: Waits for the shake routine.

In use. Use a small power, such as 0.1 to 0.25, for a machinery rumble. Its scale differs from the pixel-oriented Shake Screen action. Keep shakes short and provide a reduced-motion option for players who need it.

Shake Screen

ID: ShakeScreen | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Start the extension's screen-shake effect with horizontal, vertical or combined movement.

Field / stored key	What to enter	Starting value
Direction `direction`	Choose: both, horizontal, vertical.	"both"
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0 to 60.	0.5
Power (pixels) `power`	Number; decimals allowed. Range: 0 to 1000.	8

In use. Use a modest screen shake when a heavy mechanism locks into place. This is a different effect route and strength scale from Camera Shake. Use Stop Screen Shake to cancel an ongoing screen effect.

Stop Screen Shake

ID: StopScreenShake | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Stop the current extension-driven screen shake.

No fields in the standard catalog form.

In use. Call it if a shake is interrupted by a menu or a scene change. It does not select a camera or undo a separately running camera orbit.

Screen Fade

ID: FadeScreen | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Fade a color over the screen, or fade that cover away, over a duration.

Field / stored key	What to enter	Starting value
Direction `direction`	Choose: out, in.	"out"
Color `color`	Color picker or color string.	"#000000"
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0.0 to 60.0.	0.5

Timing: Waits for the fade routine.

In use. Fade Out to black, make the intended change, then Fade In. An Out fade leaves the cover in place; remember the In step. Use Change Scene's transition controls for a normal room change when they fit the job.

Screen Transition

ID: `ScreenTransition` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Play a screen transition effect such as dissolve, wipe, iris, pixelation or glitch.

Field / stored key	What to enter	Starting value
Transition `transitionType`	Choose: dissolve, fade, wipe, match-cut, whip-pan, iris-wipe, pixelation, glitch, comic-panel.	"dissolve"
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0.05 to 8.0.	0.8
Color `color`	Color picker or color string.	"#000000"
Direction `direction`	Choose: left, right, up, down.	"left"
Intensity `intensity`	Number; decimals allowed. Range: 0.0 to 10.0.	1.0

Timing: Waits for the transition routine.

In use. Use a short iris wipe for a stylized reveal. This action plays the effect; Change Scene is still the command that changes rooms. Direction and intensity have different visual importance for different transition types.

Set Retro Rendering Style

ID: `SetRetroStyle` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Choose the retro screen treatment: none, PSX, N64 or horror fog, with strength and pixel size.

Field / stored key	What to enter	Starting value
Retro Rendering Style `style`	Choose: none, psx, n64, horror-fog.	"psx"
Strength `strength`	Number; decimals allowed. Range: 0 to 1.	1.0
Retro Pixel Size `pixelSize`	Pixel-block size for the retro screen treatment. Range: 1 to 16.	3.0

In use. Apply a restrained PSX look to a dream, then restore none afterward. These are screen treatments; they are not a replacement for modeling geometry in a particular console style or for physical scene fog.

Zoom Blur

ID: `ZoomBlur` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Play a radial zoom blur around a normalized screen center, optionally including GUI and waiting for completion.

Field / stored key	What to enter	Starting value
Strength `strength`	Number; decimals allowed. Range: 0 to 1.	0.7
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0 to 60.	0.6
Center X (0-1) `centerX`	Normalized horizontal effect center: 0 left, 0.5 middle, 1 right. Range: 0 to 1.	0.5
Center Y (0-1) `centerY`	Normalized vertical effect center: 0 top, 0.5 middle, 1 bottom. Range: 0 to 1.	0.5
Clear Center Radius `clearRadius`	Radius around the zoom-blur center kept comparatively clear. Range: 0 to 0.8.	0.08
Direction `direction`	Choose: inward, outward.	"inward"
Include GUI in Effect `includeGUI`	Apply the screen effect to interface presentation as well.	false
Wait Until Finished `wait`	Whether this action should wait for its own effect to finish.	false

Timing: Waits when Wait is on.

In use. Use a brief outward pulse when the signal comes online. Center X/Y run from 0 to 1, so [0.5,0.5] is the middle. Keep Include GUI off when text must remain crisp.

Stop Zoom Blur

ID: `StopZoomBlur` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Cancel the active zoom-blur treatment.

No fields in the standard catalog form.

In use. Use it when a timed effect is interrupted. If a persistent shader stack also contains a blur, stopping this one-shot effect does not necessarily remove that separately authored shader.

Room changes, 2D controls and environmental state

Keep room transitions at the end of a local sequence where possible. After changing rooms, an old-room object ID may no longer resolve. Layer and depth commands apply to the 2D/hybrid scene systems, not to arbitrary 3D model transforms.

Change Scene

ID: ChangeScene | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Move the active player into another scene using a spawn and optional destination position/camera/transition settings.

Field / stored key	What to enter	Starting value
Destination Scene `sceneId`	Choose a scene ID.	-
Spawn (optional) `spawnId`	Choose an arrival spawn in the destination scene. May be blank.	Blank
XYZ when no spawn `position`	Destination/placement coordinates as [X, Y, Z].	[0.0, 0.0, 0.0]

Destination Scene is supplied by the native room/object inspector or the visual graph editor. Some shared forms show only Spawn and Position. From such a form, call a script that performs the room change. Native inspector transition/camera overrides are additional to the common schema above.

Timing: Awaits the room-change routine.

In use. Choose The Archive and the from_signal spawn. Native room/object event inspectors expose Destination Scene and transition controls. Some shared action forms omit the scene selector; call a visual or text script from those surfaces instead. A spawn is usually clearer than loose coordinates.

Switch Character

ID: `SwitchCharacter` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Change the currently controlled playable character.

Field / stored key	What to enter	Starting value
Active Player `characterId`	Choose a Character Designer ID; player is a runtime alias where supported.	Blank

Timing: Awaits the character-switch routine.

In use. Switch from Ivo to another character after confirming that character is configured as playable. The new character can have its own room, inventory, HUD and journal settings. It is not a command to swap the visible model on the same actor.

Is Current Room

ID: `IsCurrentRoom` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Test the current room through the adventure extension and return the result.

Field / stored key	What to enter	Starting value
Scene `sceneId`	Choose a scene ID.	Blank

In use. Use it from a script that reads the returned Boolean. An ordinary action list does not branch on the result. In visual scripting, choose If Current Room when you want visible True and False paths.

Set Room Dimension

ID: `SetRoomDimension` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a scene's runtime dimension setting between 3D, 2D and 2.5D, with an option to retain the 3D layer.

Field / stored key	What to enter	Starting value
Room (empty = current) `sceneId`	Choose a scene ID. May be blank.	Blank
Room Dimension `dimension`	Choose: 3d, 2d, 2.5d.	"3d"
Keep 3D Layer `keep3DLayer`	Retain the 3D layer when using the hybrid room setup.	true

In use. Use it only after the target scene has the assets and navigation appropriate to that mode. Switching a setting does not automatically build a painted backdrop, reposition every entity or convert navigation geometry.

Set 2D Camera

ID: `Set2DCamera` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set the 2D camera's position, zoom and rotation for a scene.

Field / stored key	What to enter	Starting value
Room (empty = current) `sceneId`	Choose a scene ID. May be blank.	Blank
Camera Position XY `position`	2D position as [X, Y].	[0.0, 0.0]
Camera Zoom XY `zoom`	[X, Y] numeric array.	[1.0, 1.0]
Camera Rotation (degrees) `rotation`	2D camera rotation angle; follow the field label for its units.	0.0

In use. Zoom a painted-room view to [1.25,1.25]. This acts on the 2D camera system; it does not change a 3D perspective camera's field of view. Keep both zoom values above zero.

Show / Hide 2D Layer

ID: `Set2DLayerVisible` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Show or hide a named 2D layer.

Field / stored key	What to enter	Starting value
Room (empty = current) `sceneId`	Choose a scene ID. May be blank.	Blank
2D Layer ID `layerId`	Text.	Blank
Visible `visible`	Explicit visible/hidden state.	true

In use. Hide a foreground curtain layer after it opens. Select the layer ID used by the room, not the ID of an image asset imported into the project.

Set 2D Entity Depth

ID: `Set2DEntityDepth` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change an entity's sorting depth in the 2D/hybrid scene.

Field / stored key	What to enter	Starting value
Room (empty = current) `sceneId`	Choose a scene ID. May be blank.	Blank
2D Character / Object `entityId`	Choose an entity in the selected scene.	Blank
Drawing Depth `depth`	Whole number.	100

In use. Adjust a foreground prop's depth so it appears in front of the intended sprites. Depth is drawing order; it is not a height coordinate or a collision box dimension.

Set 2D Layer Parallax

ID: `Set2DParallax` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a named 2D layer's parallax ratio.

Field / stored key	What to enter	Starting value
Room (empty = current) `sceneId`	Choose a scene ID. May be blank.	Blank
2D Layer ID `layerId`	Text.	Blank
Parallax Ratio XY `ratio`	[X, Y] numeric array.	[1.0, 1.0]

In use. Use a smaller camera-motion ratio for a distant sky and a larger one for nearby scenery. The layer must already be set up with suitable artwork and coverage; a ratio does not extend the bitmap.

Change Room Camera Bounds

ID: `SetRoomCameraBounds` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set whether camera bounds are enabled and edit the room's X/Z limits.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Bounds Enabled `enabled`	Explicit on/off state for the feature or response.	true
Left / Minimum X `minX`	Number; decimals allowed.	0
Right / Maximum X `maxX`	Number; decimals allowed.	0
Top / Minimum Z `minZ`	Number; decimals allowed.	0
Bottom / Maximum Z `maxZ`	Number; decimals allowed.	0

In use. Constrain a 3D tracking camera to the playable room. These are camera limits, not the player's navigation bounds. Expanding them alone does not make an oversized backdrop walkable.

Enable / Disable Automatic Interaction Approach

ID: `SetInteractionApproachEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set the global runtime approach-to-target behavior, including distance and whether to face the target.

Field / stored key	What to enter	Starting value
Automatic Approach Enabled `enabled`	Explicit on/off state for the feature or response.	true
Default Stop Distance `distance`	Number; decimals allowed. Range: 0.0 to 20.0.	0.35
Face Target After Approaching `faceTarget`	On/off.	true

In use. Enable walking up to interactive props, then choose a modest stop distance. If the player stops too far away, inspect both approach settings and the physical collider. The event volume is a separate object.

Set Object Interaction Approach

ID: `SetObjectInteractionApproach` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Override approach behavior for one object, or inherit the global behavior.

Field / stored key	What to enter	Starting value
Object Room `sceneId`	Choose a scene ID. May be blank.	Blank
Object / Character `entityId`	Choose an entity in the selected scene.	Blank
Approach Override `mode`	Choose: inherit, enabled, disabled.	"inherit"
Stop Distance (0 = global) `distance`	Number; decimals allowed. Range: 0.0 to 20.0.	0.0
Face Target When Override Is Enabled `faceTarget`	On/off.	true

In use. Give the door its own enabled approach mode and a close stop distance. A small distance cannot move the character through an oversized blocking collider. Set collider dimensions independently in the object's collision settings.

Set Room Brightness

ID: `SetSceneBrightness` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a scene's brightness multiplier.

Field / stored key	What to enter	Starting value
Scene `sceneId`	Choose a scene ID.	Blank
Brightness (0 = Dark, 1 = Normal) `brightness`	Number; decimals allowed.	1.0

In use. Darken the room slightly when the power fails. This is a broad rendering adjustment, not a command to move a light or create a reflective surface. Compare the result with exposure and renderer settings.

Set Room Tint

ID: `SetSceneTint` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Apply a color tint to a scene with a chosen strength.

Field / stored key	What to enter	Starting value
Scene `sceneId`	Choose a scene ID.	Blank
Tint Color `tintColor`	Color picker or color string.	"#ffffff"
Tint Strength (0 = Off, 1 = Full) `strength`	Number; decimals allowed. Range: 0.0 to 1.0.	1.0

In use. Use a pale blue tint for a cold room, then restore white or zero strength as appropriate. The tint affects presentation; it does not recolor imported source textures on disk.

Enable / Disable Flashlight Mode

ID: `SetFlashlightMode` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable or disable the flashlight-style visibility mask and adjust its strength, radius and shape.

Field / stored key	What to enter	Starting value
Flashlight Enabled `enabled`	Explicit on/off state for the feature or response.	true
Light Strength `strength`	Number; decimals allowed. Range: 0.0 to 8.0.	1.0
Light Radius `radius`	Number; decimals allowed. Range: 8.0 to 4096.0.	180.0
Light Shape `shape`	Choose: circle, spherical, square, rectangle, triangle.	"circle"

In use. Use a circular mask for a dark inspection scene. Radius is a screen-oriented value in this feature; do not confuse it with an OmniLight3D range. This is not a replacement for a physical light attached to a character's hand.

Set Snow Accumulation

ID: `SetSnowAccumulation` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set or animate accumulated snow on the scene, with duration/pace and optional waiting.

Field / stored key	What to enter	Starting value
Scene (empty = current) `sceneId`	Choose a scene ID. May be blank.	Blank
Accumulation Amount (0 to 1) `amount`	Target accumulated snow amount from 0 to 1. Range: 0.0 to 1.0.	1.0
Transition Duration (seconds, 0 = authored pace) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0.0 to 86400.0.	0.0
Amount Per Second (used only when duration = 0) `pace`	Snow accumulation/melt pace; use the duration route for a simple timed test. Range: 0.0 to 100.0.	0.0
Wait Until Accumulation Finishes `waitForCompletion`	Wait for the snow animation to complete before continuing.	true

Timing: Waits when Wait for Completion is on.

In use. Build snow over a short time while a reveal runs. Use Wait for Completion only when subsequent instructions should be delayed. The target room must have snow-capable surfaces and appropriate visual settings.

Melt Accumulated Snow

ID: `MeltSnow` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Reduce the scene's accumulated snow over time.

Field / stored key	What to enter	Starting value
Scene (empty = current) `sceneId`	Choose a scene ID. May be blank.	Blank
Melt Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0.0 to 86400.0.	2.0
Amount Per Second (used only when duration = 0) `pace`	Snow accumulation/melt pace; use the duration route for a simple timed test. Range: 0.0 to 100.0.	0.0
Wait Until Melting Finishes `waitForCompletion`	Wait for the snow animation to complete before continuing.	true

Timing: Waits when Wait for Completion is on.

In use. Melt the snow after activating a heater. Match duration/pace to the intended effect and test which surfaces receive snow. It changes accumulated snow, not the count of particles emitted by a separate falling-snow effect.

Continuous object effects and attachments

Configure an effect or attach point first, then use these actions to control it. An attach point is a named location on an object, model bone or sprite. Contact watches report proximity between points; they are not a full physics collision system.

Toggle Motion

ID: `ToggleMotion` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Invert an object's authored motion-enabled state, optionally mirroring that state into a variable.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Object `entityId`	Choose an entity in the selected scene.	Blank
State Variable (optional) `variable`	Choose a game variable name. May be blank.	Blank

In use. Toggle a rotating sign when a switch is used. Configure the object's motion in its inspector first. Repeated events invert the state again; use Set Motion when a specific final state matters.

Set Motion

ID: `SetMotion` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Explicitly enable or disable authored object motion.

Field / stored key	What to enter	Starting value
Room `sceneId`	Choose a scene ID.	Blank
Object `entityId`	Choose an entity in the selected scene.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true
State Variable (optional) `variable`	Choose a game variable name. May be blank.	Blank

In use. Start the clock's rotating mechanism after power is restored. Choose Enabled on for a repeat-safe command. This does not create the rotation path, speed or scaling motion; it controls the setup already present on the object.

Toggle Glow

ID: `ToggleGlow` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Invert an object's authored glow state and optionally record it in a variable.

Field / stored key	What to enter	Starting value
Entity `entityId`	Choose a scene entity ID.	Blank
Variable `variable`	Text.	Blank

In use. Toggle a test lamp during inspection. Visible bloom also depends on renderer/environment support. A bright material without bloom is still emissive; glow is not a mirror reflection of the player.

Set Glow

ID: `SetGlow` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Set an object's authored glow state explicitly.

Field / stored key	What to enter	Starting value
Entity `entityId`	Choose a scene entity ID.	Blank
Enabled `enabled`	Explicit on/off state for the feature or response.	true
Variable `variable`	Text.	Blank

In use. Enable the signal lamp once the circuit is powered. Use the object's glow setup and the room's bloom/glow settings together. Repeating Set Glow Enabled is safer than toggling when a trigger can fire repeatedly.

Toggle Rgb String

ID: `ToggleRGBString` | Shared list / GUI · Visual graph · Text action wrapper · Timeline

Toggle an authored RGB light string, optionally writing its state to a variable.

Field / stored key	What to enter	Starting value
Entity `entityId`	Choose a scene entity ID.	Blank
Variable `variable`	Text.	Blank

In use. Turn decorative bulbs on after a celebration begins. The bulbs, spacing and light settings belong to the RGB string setup; this action does not create a new string from an ordinary mesh.

Set Attach Point

ID: `SetAttachPoint` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Create or update a named attach point with position, rotation, coordinate space, bone and contact radius.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Position `position`	Destination/placement coordinates as [X, Y, Z].	[0, 0, 0]
Rotation (degrees) `rotation`	Rotation values as a numeric array; follow the field's stated units.	[0, 0, 0]
Bone Name (optional) `bone`	Exact imported skeleton bone name, if this point follows a bone.	Blank
Coordinate Space `space`	Choose: object, model, sprite_uv.	"object"
Contact Radius `contactRadius`	Radius used to test contact around an attach point.	0.1
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Define right_hand_grip on a character's hand bone. Object/model coordinates and sprite UV coordinates mean different things; choose the space deliberately. The bone name must match the imported skeleton exactly.

Remove Attach Point

ID: `RemoveAttachPoint` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Remove a named attach point from an object or character.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank

In use. Remove a temporary alignment point after staging a sequence. Any attachment or watch referring to that name must be cleaned up as well; it cannot follow a point that no longer exists.

Get Attach Point

ID: `GetAttachPoint` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Read an attach point's information and optionally store the result in a game variable.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Store Result in Variable `resultVariable`	Name of the game variable that receives the returned value.	Blank

In use. Store a hand point in hand_info for a script to inspect. This returns structured information rather than a ready-made visual branch. Inspect the result before assuming a missing point has a usable world position.

Attach Points

ID: `AttachPoints` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Attach one entity's point to another entity's point, optionally aligning rotation and suspending movement.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Target Object `targetId`	Choose a scene entity ID.	Blank
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	Blank
Align Rotation `alignRotation`	Align point orientation as well as position.	true
Suspend Movement `suspendMovement`	Suspend the attached character/entity movement where supported.	true

In use. Attach a prop grip point to the player's hand grip point. Choose which entity is the child and which is the target. Suspending a character's movement is useful for staging but must fit the rest of the sequence.

Align Attach Points

ID: `AlignAttachPoints` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Align two attach points once without establishing the same continuing attachment relationship.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Target Object `targetId`	Choose a scene entity ID.	Blank
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	Blank
Align Rotation `alignRotation`	Align point orientation as well as position.	true

In use. Align two hands before a handshake pose. If either character then moves or the animation moves the hands apart, a one-time alignment is not guaranteed to keep contact throughout the animation.

Attach Points Touching

ID: `AttachPointsTouching` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Check whether two points are touching within their radii and tolerance, and optionally store the result.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Target Object `targetId`	Choose a scene entity ID.	Blank
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	Blank
Contact Tolerance `tolerance`	Extra allowed contact distance; use a nonnegative number.	0.0
Store Result in Variable `resultVariable`	Name of the game variable that receives the returned value.	Blank

In use. Store hand_contact, then branch on it in a graph or script. Putting this test in a plain list does not make later instructions conditional. Use If Attach Points Touching for visible graph branches.

Get Attach Point Contacts

ID: `GetAttachPointContacts` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Get contact information for the requested point/target filters.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Target Object `targetId`	Choose a scene entity ID.	Blank
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	Blank
Contact Tolerance `tolerance`	Extra allowed contact distance; use a nonnegative number.	0.0
Store Result in Variable `resultVariable`	Name of the game variable that receives the returned value.	Blank

In use. Read contact details for a carried tool and let a script decide which target was touched. A returned collection or record is not automatically converted into a puzzle success flag.

Watch Attach Point Contacts

ID: `WatchAttachPointContacts` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Start a named contact watch, optionally using swept detection between frames.

Field / stored key	What to enter	Starting value
Watch ID `watchId`	Stable name for this continuing contact watch.	Blank
Object / Character `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Target Object `targetId`	Choose a scene entity ID.	Blank
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	Blank
Contact Tolerance `tolerance`	Extra allowed contact distance; use a nonnegative number.	0.0
Detect Between Frames `swept`	Check movement between samples to catch brief contacts.	true

In use. Watch tool_touch during a fast tool animation. Listen for OnAttachPointContact, filter the watch ID and the relevant point IDs, and stop the watch when the animation or puzzle phase ends.

Stop Attach Point Watch

ID: `StopAttachPointWatch` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Stop a previously started contact watch by its watch ID.

Field / stored key	What to enter	Starting value
Watch ID `watchId`	Stable name for this continuing contact watch.	Blank

In use. Stop tool_touch after the tool has been put away. Use the same watch ID that started it. Removing the visible prop alone is not a clear lifecycle plan for a continuing watch.

Attach Object

ID: `AttachObject` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Attach an object to a target object's attach point.

Field / stored key	What to enter	Starting value
Object `entityId`	Choose a scene entity ID.	Blank
Attach Point `point`	Name of the source attach point.	Blank
Target Object `targetId`	Choose a scene entity ID.	Blank
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	Blank

In use. Attach a lantern to the player's hand. Use the prop's source point and the target's grip point when available. Giving an inventory lantern and attaching a visible lantern are separate operations.

Detach Object

ID: `DetachObject` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Detach an object that was attached through the attachment system.

Field / stored key	What to enter	Starting value
Object `entityId`	Choose a scene entity ID.	Blank

In use. Detach the lantern before placing it on the table, then position it as required. Detaching does not automatically give an inventory item back or select a new animation.

Attach Particle Effect

ID: `AttachParticleEffect` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Attach an existing particle-effect entity to a target point with an optional offset.

Field / stored key	What to enter	Starting value
Particle Effect `entityId`	Choose a scene entity ID.	Blank
Target Object `targetId`	Choose a scene entity ID.	Blank
Target Attach Point `targetPoint`	Name of the attach point on the target entity.	Blank
Attachment Offset `offset`	Attachment offset as [X, Y, Z].	[0, 0, 0]

In use. Attach smoke to a pipe's smoke_outlet point. Select the scene's particle entity, not just an imported texture used by that emitter. Keep the offset in the intended attachment space.

Detach Particle Effect

ID: `DetachParticleEffect` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Detach a particle-effect entity from its target.

Field / stored key	What to enter	Starting value
Particle Effect `entityId`	Choose a scene entity ID.	Blank

In use. Detach a trailing effect when the carried object is put down. If the emitter should stop too, change its emission/enabled state through the appropriate effect setup or supported property action.

GUIs, element state and data-driven widgets

GUI targets have two levels: the interface ID and an element ID inside it. Use the interface-level commands for the whole window and element-level commands for one button, label or widget. Visual Designer preview is not the place to test event execution.

Show Gui

ID: `ShowGUI` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Show an authored interface.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank

In use. Show a note-reading panel after the player examines a document. Check its runtime activation, layer and blocking settings. Showing it is different from enabling a disabled interface.

Hide Gui

ID: `HideGUI` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Hide an authored interface.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank

In use. Hide the note panel on its Close button. It does not erase its definition or necessarily clear the story variables behind it. If another activation rule immediately shows it again, fix that rule rather than adding repeated Hide calls.

Enable Gui

ID: `EnableGUI` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable an authored interface through the runtime GUI manager.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank

In use. Enable a menu after a sequence that disabled it. Whether it is currently visible is a separate state; use Show GUI as well if it needs to appear.

Disable Gui

ID: `DisableGUI` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Disable an authored interface through the runtime GUI manager.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank

In use. Prevent interaction with an interface while a transition is running. Use Hide GUI if the intention is to remove it from view. Test the final disabled appearance in the running game.

Show Gui Element

ID: `ShowGUIElement` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Show one element inside an authored GUI.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Element / Group `elementId`	Choose an element inside the selected GUI.	Blank

In use. Reveal an archive tab button after the player finds the key. Select both the owning GUI and that element. The parent GUI and any parent panel must also be visible.

Hide Gui Element

ID: `HideGUIElement` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Hide one element inside an authored GUI.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Element / Group `elementId`	Choose an element inside the selected GUI.	Blank

In use. Hide a temporary objective label after completion. A hidden child cannot be made visible merely by showing its parent if its own visible flag remains off.

Enable Gui Element

ID: `EnableGUIElement` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable one element for interaction.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Element / Group `elementId`	Choose an element inside the selected GUI.	Blank

In use. Enable the Continue button once the necessary option has been selected. A parent interface or modal input lock can still prevent access even when this child's enabled flag is on.

Disable Gui Element

ID: `DisableGUIElement` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Disable one element for interaction.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Element / Group `elementId`	Choose an element inside the selected GUI.	Blank

In use. Disable Save until a save slot has been selected. Decide what visual disabled treatment the element should use so the player understands why it cannot be used.

Change Gui Property

ID: `ChangeGUIProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a supported property on an authored GUI.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Property `property`	Choose a GUI property.	Blank
To `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Change a window's runtime appearance or layout using a property offered by the picker. Match the value type to the property. This is a runtime edit, not a request to rewrite the editor's theme.

Change Gui Element Property

ID: `ChangeGUIElementProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a supported property on a GUI element.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Element / Group `elementId`	Choose an element inside the selected GUI.	Blank
Property `property`	Choose an element property.	Blank
To `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Change the status label's text after the signal is powered. Use the correct property key and value type. In a GUI event's script context, elementId this can refer to the calling element; a global event usually has no such caller.

List Box: Add Item

ID: `ListBoxAddItem` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Add a manual list-box item with an ID, text, optional icon, disabled state and insertion index.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
List Box `elementId`	Choose an element inside the selected GUI.	Blank
New Item ID `itemId`	Text.	Blank
Item Text `text`	Text.	Blank
Optional Icon `iconAssetId`	Choose an imported image. May be blank.	Blank
Disabled `disabled`	On/off.	true
Insert Index (-1 = End) `insertIndex`	Zero-based insertion position; -1 appends to the list.	-1

In use. Add a discovered location to a travel list with a stable ID. An insertion index of -1 appends. Turn Disabled off when the player should be able to select the row. For list boxes bound to an array or inventory, change the data source rather than expecting manual entries to override it permanently.

List Box: Delete Item

ID: `ListBoxDeleteItem` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Delete a list-box item by its item ID.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
List Box `elementId`	Choose an element inside the selected GUI.	Blank
Item ID `itemId`	Text.	Blank

In use. Remove a destination that is no longer available. The item's ID is not its visible label or a row number. Consider disabling it instead when the player should still see that it exists.

List Box: Modify Item

ID: `ListBoxModifyItem` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change an existing list item's text, ID, icon, disabled or selected property.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
List Box `elementId`	Choose an element inside the selected GUI.	Blank
Item ID `itemId`	Text.	Blank
Property `property`	Choose: text, id, iconAssetId, disabled, selected.	"text"
New Value `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Disable a travel destination during a storm. Changing an item ID can break later lookups, so treat IDs as stable references whenever possible.

List Box: Clear Items

ID: `ListBoxClearItems` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Clear the manual items in a list box.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
List Box `elementId`	Choose an element inside the selected GUI.	Blank

In use. Empty a results list before rebuilding it. An array- or inventory-backed list may repopulate from its source; clear or change that source when appropriate.

Carousel: Add Image

ID: `CarouselAddImage` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Add an imported image asset to an image carousel.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Image Carousel `elementId`	Choose an element inside the selected GUI.	Blank
Image Asset `assetId`	Choose an imported image.	Blank

In use. Add a newly discovered photograph to a gallery. Choose the GUI and carousel element, then the image asset. This is not a command to create a new image file.

Carousel: Remove Image

ID: `CarouselRemoveImage` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Remove a selected image asset from a carousel's authored/runtime list.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Image Carousel `elementId`	Choose an element inside the selected GUI.	Blank
Image Asset `assetId`	Choose an imported image.	Blank

In use. Remove a temporary clue image when the inspection ends. Check what happens if it was the current slide, and test the first/last-image behavior.

Carousel: Clear Images

ID: `CarouselClearImages` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Clear a carousel's image list.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Image Carousel `elementId`	Choose an element inside the selected GUI.	Blank

In use. Reset a manual inspection gallery before filling it with another object's images. Inventory-driven carousels can obtain their contents from inventory instead; update the appropriate data source.

Carousel: Previous Image

ID: `CarouselPrevious` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Move a carousel to its previous image.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Carousel `elementId`	Choose an element inside the selected GUI.	Blank

In use. Wire a left-arrow button to the chosen carousel. Wrapping, animation and layout remain properties of that carousel.

Carousel: Next Image

ID: `CarouselNext` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Move a carousel to its next image.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Carousel `elementId`	Choose an element inside the selected GUI.	Blank

In use. Wire a right-arrow button to the chosen carousel. This navigates the widget, rather than advancing dialogue or changing scenes.

Save Game Widget: Command

ID: `SaveGameWidgetCommand` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Control a save widget's navigation, selection or selected/explicit-slot operations.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
Save Game Widget `elementId`	Choose an element inside the selected GUI.	Blank
Command `command`	Choose: previous, next, up, down, refresh, select-first, select-previous, select-next, load-selected, save-selected, rename-selected, delete-selected, select-slot, load-slot, save-slot, rename-slot, delete-slot.	"previous"
Explicit Slot (-1 = Selected) `slot`	Explicit save slot number. Selected-slot widget commands use widget selection instead.	-1

In use. Use select-slot with a slot number, or save-selected after the player picks a row. Commands ending in -selected use widget selection; commands ending in -slot use Slot. Test overwrite, rename and delete behavior on disposable saves.

Center GUI Model

ID: `CenterGUIModel` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Fit and center the model displayed inside a GUI 3D model viewer.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
3D Model `elementId`	Choose an element inside the selected GUI.	Blank

In use. Offer a Reset View button after the player rotates a model. It targets the GUI viewer, not the scene's active gameplay camera or the model's source file.

Play GUI Model Animation

ID: `PlayGUIModelAnimation` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Play a named animation in a GUI model viewer, with loop and speed controls.

Field / stored key	What to enter	Starting value
GUI `guiId`	Choose an authored GUI ID.	Blank
3D Model `elementId`	Choose an element inside the selected GUI.	Blank
Animation `animation`	Text.	Blank
Loop `loop`	On/off.	true
Speed `speed`	Playback speed multiplier; 1 is normal speed.	1.0

In use. Play an idle loop in a character inspection window. Match the imported animation name. A model with no animation clips cannot gain one just by entering a name here.

Control profiles, verbs and pointer behavior

These commands affect how the player operates the game. Movement profile, interaction mode, input device and visibility of the pointer are separate settings. Provide a way back when an experiment changes the controls.

Set Runtime Controls Enabled

ID: `SetRuntimeControlsEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Enable or disable the runtime control-profile system.

Field / stored key	What to enter	Starting value
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Enable the newer adaptive controls for a sequence that uses device switching. Disabling the profile system is not identical to Set Player Input Enabled false; it changes which control machinery is in use.

Set Movement Profile

ID: `SetMovementProfile` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Choose legacy, point-and-click, direct, first-person or drag-to-move movement.

Field / stored key	What to enter	Starting value
Movement Profile `profile`	Choose: legacy, point-and-click, direct, first-person, drag-to-move.	"point-and-click"

In use. Switch to direct movement for a short action scene and restore point-and-click afterward. Navigation, camera and input mappings still need to suit the chosen profile.

Set Interaction Mode

ID: `SetInteractionMode` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Choose the runtime hotspot interaction style, including context, verb, direct, cycling cursor and verb-coin variants.

Field / stored key	What to enter	Starting value
Interaction Mode `mode`	Choose: context, verb-single-click, verb-double-click, verb-direct, verb-direct-double-click, direct, cycling-cursor, verb-coin, verb-coin-first.	"context"

In use. Use verb-coin for a compact interaction menu. Changing the mode does not create missing verb responses on objects. Test empty-ground clicks, inventory combinations and the device the player is actually using.

Automatic Input Switching

ID: `SetAutoInputSwitching` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Enable or disable automatic input-device switching.

Field / stored key	What to enter	Starting value
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Turn it on for a game that should adapt between mouse and gamepad. Turn it off during a fixed-device control demonstration. Device detection and movement-profile switching are related settings, not the same switch.

Set Input Device

ID: `SetInputDevice` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Select auto, mouse-keyboard, gamepad or touch input mode.

Field / stored key	What to enter	Starting value
Input Device `device`	Choose: auto, mouse-keyboard, gamepad, touch.	"auto"

In use. Choose Auto for normal play or force Gamepad while testing controller prompts. Forcing a device is not a substitute for having usable bindings and an actual controller available.

Proximity Interaction

ID: `SetProximityInteractionEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Enable or disable nearest/proximity-based interaction.

Field / stored key	What to enter	Starting value
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Enable it for controller play in a small room. Check proximity radius, hotspot eligibility and line-of-play expectations so the nearest usable target is the one the player expects.

Set Control Setting

ID: `SetControlSetting` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Change one supported runtime control setting with a value of the matching type.

Field / stored key	What to enter	Starting value
Setting `setting`	Choose: enabled, cursorAdaptation, cyclingEnabled, movementProfile, autoSwitchInput, inputDevice, autoSwitchMovement, mouseProfile, gamepadProfile, touchProfile, proximityInteraction, proximityRadius, showHints, controllerIcons, coinEnabled, dragThreshold, gamepadLookSpeed.	"coinEnabled"
Value `value`	A value of the type required by the selected variable, operation or property.	true

In use. Set proximityRadius to a number or coinEnabled to a Boolean. The same Value box accepts different types depending on Setting; the text "false" is not the same as Boolean false.

Get Control State

ID: `GetControlState` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Read the current control-profile state and store it in a game variable.

Field / stored key	What to enter	Starting value
Result Variable `resultVariable`	Name of the game variable that receives the returned value. Custom values are accepted.	Blank

In use. Store controls_before before entering an inspection mode and inspect the returned state in a script. The result is structured state, not a single enabled flag or an automatic restore command.

Open Interaction Coin

ID: `OpenInteractionCoin` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Open the interaction coin for a target entity when that feature is configured.

Field / stored key	What to enter	Starting value
Object / Character `entityId`	Choose a scene entity ID. May be blank.	Blank

In use. Open a coin from a dedicated Inspect button or a hotspot interaction. The target must have useful verbs. A coin opened with no meaningful target cannot invent responses.

Close Interaction Coin

ID: `CloseInteractionCoin` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Close the currently open interaction coin.

No fields in the standard catalog form.

In use. Close the coin when a menu takes over. It hides the coin, rather than disabling the feature for the rest of the game.

Cycle Verb

ID: `CycleInteractionVerb` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Move forward or backward through the runtime verb cycle.

Field / stored key	What to enter	Starting value
Direction `direction`	Whole number. Range: -1 to 1.	1

In use. Use direction 1 for the next verb and -1 for the previous one. A value of 0 does not mean "first verb." The cycle depends on the enabled verbs and current interaction settings.

Interact with Nearest Hotspot

ID: `InteractNearest` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Perform a verb interaction with the nearest eligible hotspot under the proximity system.

Field / stored key	What to enter	Starting value
Verb `verbId`	Choose a Verb Designer ID. May be blank.	Blank

In use. Use it for a controller interaction button. Select a verb or leave the optional verb selection to the current behavior. Test target selection in a crowded room.

Enable Mouse Clicks

ID: `SetMouseClicksEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Enable or disable the extension's mouse-click interaction switch.

Field / stored key	What to enter	Starting value
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable scene clicks while a special control sequence owns the pointer. It is a different switch from hiding the cursor. Restore it on every completion and cancellation path.

Enable Mouse Pointing

ID: `SetMousePointingEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Enable or disable mouse pointing/hover behavior through the extension.

Field / stored key	What to enter	Starting value
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable hover targeting during a locked cinematic view. This does not necessarily remove every GUI interaction or keyboard route; use the appropriate broader input lock when needed.

Show Mouse Pointer

ID: `SetMousePointerVisible` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Show or hide the mouse pointer.

Field / stored key	What to enter	Starting value
Visible `visible`	Explicit visible/hidden state.	true

In use. Hide it for a cinematic, then show it when play returns. A hidden cursor can still leave other controls active, so do not use this as an input lock.

Change Current Verb Command

ID: `ChangeVerb` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Select a verb from the project's Verb Designer.

Field / stored key	What to enter	Starting value
New Verb `verbId`	Choose a Verb Designer ID.	Blank

In use. Select Look when an inspection interface opens. The verb ID is stored even when the display name is translated or renamed. Choosing the verb alone does not execute it on an object.

Enable / Disable Verb Command Usage

ID: `SetVerbUsageEnabled` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Enable or disable verb usage at runtime.

Field / stored key	What to enter	Starting value
Enabled `enabled`	Explicit on/off state for the feature or response.	true

In use. Disable normal verb selection while a special puzzle interface is active, then restore it. It does not remove authored verb definitions or their response lists.

Change Game Cursor

ID: `ChangeGameCursor` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change the runtime cursor to an authored cursor definition.

Field / stored key	What to enter	Starting value
Cursor `cursorId`	Choose a Cursor Designer ID.	Blank

In use. Use a magnifier cursor for inspection. Hover feedback and interaction-mode logic may select other cursor states, so test the change while pointing at both ground and hotspots.

Audio, video, FMV and recorded gameplay

Sound effects, music, voice and video use separate playback routes. A sound channel gives you a handle for stopping, pausing or fading a group of effects. Starting playback usually lets the next instruction run immediately.

Play Sound

ID: `PlaySound` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Start an imported audio asset as a sound effect, optionally on a named channel.

Field / stored key	What to enter	Starting value
Sound Asset `assetId`	Choose an imported audio asset.	Blank
Channel / ID `channel`	Sound-effect channel name. Reuse it to pause, stop or fade that sound group.	Blank

In use. Play chime on puzzle, then continue the list. Use that same channel to stop or fade the effect later. A blank channel has broader/default behavior; name important continuing sounds.

Stop Sound

ID: `StopSound` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Stop effects selected by channel.

Field / stored key	What to enter	Starting value
Channel / ID `channel`	Sound-effect channel name. Reuse it to pause, stop or fade that sound group.	Blank

In use. Stop the machine channel when the power is cut. Use a specific channel so unrelated effects can continue. This does not stop music or voice.

Sound: Pause

ID: PauseSound | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Pause effects selected by channel.

Field / stored key	What to enter	Starting value
Channel / ID (empty = all) `channel`	Sound-effect channel name. Reuse it to pause, stop or fade that sound group. May be blank.	Blank

In use. Pause a repeating machine sound during an inspection that freezes its motion. Pausing is reversible; stopping ends playback instead.

Sound: Resume

ID: ResumeSound | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Resume paused effects selected by channel.

Field / stored key	What to enter	Starting value
Channel / ID (empty = all) `channel`	Sound-effect channel name. Reuse it to pause, stop or fade that sound group. May be blank.	Blank

In use. Resume the machine channel together with the machine's motion. It resumes existing players; it does not choose or reload an audio asset.

Sound: Fade

ID: FadeSound | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Fade a sound channel to a normalized volume over time, waiting for the fade routine.

Field / stored key	What to enter	Starting value
Channel / ID (empty = all) `channel`	Sound-effect channel name. Reuse it to pause, stop or fade that sound group. May be blank.	Blank
Target Volume `volume`	Normalized target volume: 0 is silent, 1 is full volume. Range: 0.0 to 1.0.	0.0
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0.0 to 60.0.	1.0

Timing: Waits for the fade routine.

In use. Fade ambience to 0 over 1.5 seconds before stopping it. Volume uses 0 to 1, not decibels. A final volume near zero stops the selected sound players.

Play Music

ID: `PlayMusic` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Start the chosen audio asset as music.

Field / stored key	What to enter	Starting value
Music Asset `assetId`	Choose an imported audio asset.	Blank
Skippable `skippable`	On/off.	true

In use. Start a room theme and let the following actions continue. The catalog includes Skippable, but the current action bridge does not use that field to implement music skipping; do not build a skip button around it.

Stop Music

ID: `StopMusic` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Stop the current music track.

No fields in the standard catalog form.

In use. Stop the room theme before a silent reveal. It is music-specific and leaves separate effects and voice alone.

Music: Pause

ID: `PauseMusic` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Pause the current music track.

No fields in the standard catalog form.

In use. Pause music while an in-game device temporarily takes over the soundtrack. The track position is kept for Resume Music.

Music: Resume

ID: `ResumeMusic` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Resume the paused music track.

No fields in the standard catalog form.

In use. Use it after the device view closes. If another action replaced the track, the old track is not automatically restored.

Music: Fade

ID: `FadeMusic` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Fade the current music to a normalized volume, waiting for the fade routine.

Field / stored key	What to enter	Starting value
Target Volume `volume`	Normalized target volume: 0 is silent, 1 is full volume. Range: 0.0 to 1.0.	0.0
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits. Range: 0.0 to 60.0.	1.0

Timing: Waits for the fade routine.

In use. Fade to 0 over 2 seconds for a room ending, or to a quieter nonzero level under narration. A zero target stops playback; volume is 0 to 1.

Skip Ahead in Soundtrack

ID: `SeekMusic` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Move playing music to minutes plus seconds.

Field / stored key	What to enter	Starting value
Minute `minutes`	Whole minutes; added to Seconds to form a playback position.	0
Second `seconds`	Number; decimals allowed.	0

In use. Seek to 1 minute 10 seconds to check a cue. This is media playback positioning, not a timeline scheduler or a music crossfade.

Stop Startup Splash Audio

ID: `StopStartupSplashAudio` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Stop audio still playing from startup splash presentation.

No fields in the standard catalog form.

In use. Use it when the first playable scene should take over the soundtrack. It does not disable splash screens or erase the splash audio asset.

Play Video

ID: `PlayVideo` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Start an imported video with full-screen, skippable and muted options.

Field / stored key	What to enter	Starting value
Video Asset `assetId`	Choose an imported video.	Blank
Full Screen `fullScreen`	Use the video's full-screen presentation route.	true
Skippable `skippable`	On/off.	true
Muted `muted`	Play the video without its audio.	true

In use. Show a short reel, then use OnVideoEnd if later logic depends on natural completion. Starting the video is not a promise that the next action waits for the whole clip.

Stop Video

ID: `StopVideo` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Stop the current video playback.

No fields in the standard catalog form.

In use. Use it to cancel a video inspection. Test the cancel path separately from natural completion; an explicit stop should not be assumed to raise the same completion event.

Seek Video

ID: `SeekVideo` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Seek the current video to minutes plus seconds.

Field / stored key	What to enter	Starting value
Minute `minutes`	Whole minutes; added to Seconds to form a playback position.	0
Second `seconds`	Number; decimals allowed.	0

In use. Seek to 0 minutes 12 seconds while testing a clip. Playback and seeking depend on the imported format and decoder; test the build you intend to distribute.

Start FMV

ID: `StartFMV` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Start the extension's FMV playback route with an end, point or interrupt continuation mode and optional script.

Field / stored key	What to enter	Starting value
Movie Asset ID `assetId`	Choose an imported video.	Blank
Playback Limit `mode`	Choose: end, point, interrupt.	"end"
Play Until (seconds) `endTime`	FMV point-mode endpoint; this differs from the seek destination. Range: 0 to 86400.	0
Completion Script `scriptId`	Choose a visual or supported text script. May be blank.	Blank

In use. Play until a chosen point, then call the script that presents the next interactive decision. Configure the video and continuation intentionally. FMV is a media-driven interaction route, not the Cutscene Timeline itself.

Stop FMV

ID: `StopFMV` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Stop the current FMV playback route.

No fields in the standard catalog form.

In use. Use it when leaving an FMV sequence entirely. Decide what room, interface or script takes over after cancellation.

Pause FMV

ID: `PauseFMV` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Pause the current FMV route.

No fields in the standard catalog form.

In use. Pause the clip while a decision is visible. This does not pause every other game system automatically.

Resume FMV

ID: ResumeFMV | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Resume FMV with the chosen continuation mode, endpoint and script.

Field / stored key	What to enter	Starting value
Playback Limit `mode`	Choose: end, point, interrupt.	"end"
Play Until (seconds) `endTime`	FMV point-mode endpoint; this differs from the seek destination. Range: 0 to 86400.	0
Completion Script `scriptId`	Choose a visual or supported text script. May be blank.	Blank

In use. Continue after the player answers a timed question. Recheck endTime and the chosen script so resuming does not immediately reach the same stop point again.

Seek FMV

ID: SeekFMV | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Move FMV to a time and set its continuation behavior.

Field / stored key	What to enter	Starting value
Time `time`	FMV seek destination in seconds. Range: 0 to 86400.	0
Playback Limit `mode`	Choose: end, point, interrupt.	"end"
Play Until (seconds) `endTime`	FMV point-mode endpoint; this differs from the seek destination. Range: 0 to 86400.	0
Completion Script `scriptId`	Choose a visual or supported text script. May be blank.	Blank

In use. Jump to a branch of an interactive film. Use Time for the seek destination and End Time for point-mode continuation; they describe different moments.

Start Gameplay Recording

ID: `StartGameplayRecording` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Begin the extension's gameplay-recording route using a writable JSON path.

Field / stored key	What to enter	Starting value
Recording Path `path`	Writable recording-data path, such as user://walkthrough.json.	"user://walkthrough.json"

In use. Record a repeatable walk through a puzzle for testing. This creates gameplay data, not a finished MP4 screen recording with sound. Keep the path writable in exported builds.

Stop Gameplay Recording

ID: `StopGameplayRecording` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Stop the active gameplay recording.

No fields in the standard catalog form.

In use. End the test run cleanly before starting another recording. It does not stop a separate operating-system video capture.

Play Gameplay Recording

ID: `PlayGameplayRecording` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Play a saved gameplay recording at the chosen speed.

Field / stored key	What to enter	Starting value
Recording Path `path`	Writable recording-data path, such as user://walkthrough.json.	"user://walkthrough.json"
Playback Speed `speed`	Playback speed multiplier; 1 is normal speed.	1

In use. Replay user://walkthrough.json while checking a room. The recording needs a compatible game state and scene setup; it is not a self-contained replacement for the project and its assets.

Stop Gameplay Playback

ID: `StopGameplayPlayback` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Stop playback of recorded gameplay data.

No fields in the standard catalog form.

In use. Use it before handing control back to the player or starting another test run. Check the active camera and input state after stopping a scripted replay.

Quests and objectives

Quest actions target definitions from the Quest Designer. Quest, stage and objective IDs form a hierarchy. Use the pickers in that order so a valid objective is selected within the intended stage.

Start Quest

ID: `StartQuest` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Start a quest and optionally select its initial stage.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Initial Stage (optional) `stageId`	Choose a stage in that quest. May be blank.	Blank

In use. Start Restore the Signal when the player accepts the task. A quest definition must already exist. This action does not create a journal widget or force a hidden journal interface onto the screen.

Quest: Set Stage

ID: `SetQuestStage` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Move a quest to a chosen stage.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Stage `stageId`	Choose a stage in that quest.	Blank

In use. Advance from Restore Power to Enter the Archive after the lever succeeds. Check whether earlier objectives should be completed before moving on, rather than assuming the stage change supplies all the story bookkeeping.

Quest: Complete Objective

ID: `CompleteObjective` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Complete a named objective, optionally selecting a stage first.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Stage `stageId`	Choose a stage in that quest.	Blank
Objective `objectiveId`	Choose an objective in that stage.	Blank

In use. Complete power inside the restore stage after the signal turns on. A supplied stage ID is applied before the objective operation, so choose it deliberately.

Quest: Fail Objective

ID: `FailObjective` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Mark an objective as failed, optionally selecting its stage first.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Stage `stageId`	Choose a stage in that quest.	Blank
Objective `objectiveId`	Choose an objective in that stage.	Blank

In use. Fail a timed optional objective when its timer expires. Keep failure consequences in a separate clear branch so a failed side objective does not accidentally end the whole quest.

Quest: Reset Objective

ID: `ResetObjective` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Reset one objective to its unfinished state, optionally selecting its stage first.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Stage `stageId`	Choose a stage in that quest.	Blank
Objective `objectiveId`	Choose an objective in that stage.	Blank

In use. Reset a repeatable practice task. It does not automatically restore consumed items, move characters or undo the rest of the room state.

Quest: Complete Quest

ID: `CompleteQuest` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Mark the quest completed.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank

In use. Complete Restore the Signal after the archive has been reached. Award score or items as separate guarded actions. Completing a quest is not automatic proof that a reward has not already been given.

Quest: Fail Quest

ID: `FailQuest` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Mark the quest failed.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank

In use. Fail a side quest after its story window closes. Provide a readable journal entry or message explaining the result; the state change alone may not tell the player what happened.

Quest: Reset Quest

ID: `ResetQuest` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Reset the quest state through the quest manager.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank

In use. Restart a repeatable challenge. Reconcile its room state, inventory and timers as well; resetting the journal state does not rebuild the whole puzzle.

Quest: Hide / Unhide

ID: `SetQuestHidden` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set a quest's hidden state.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Hidden `hidden`	Whether the quest or player visual should be hidden for this action.	true

In use. Reveal a secret quest once its first clue is found. Hidden controls journal visibility policy, rather than completing or deleting the quest.

Quest: Track / Untrack

ID: `SetQuestTracked` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set whether a quest is tracked.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Tracked `tracked`	On/off.	true

In use. Track the active investigation after the player accepts it. The GUI must contain a suitable tracked-quest display if that state is meant to be visible.

Quest: Change Property

ID: `ChangeQuestProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a supported quest property such as title, description, category or flags.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Quest Property `property`	Choose a quest property.	Blank
New Value `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Replace the quest description after a revelation. Choose a Boolean for Boolean properties and text for text properties; use dedicated completion actions for progression.

Quest: Change Stage Property

ID: `ChangeQuestStageProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a supported property's value on one quest stage.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Stage `stageId`	Choose a stage in that quest.	Blank
Stage Property `property`	Choose a stage property.	Blank
New Value `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Update a stage's descriptive text as new evidence is found. This edits presentation/data; Set Quest Stage is the command that makes a different stage current.

Quest: Change Objective Property

ID: `ChangeQuestObjectiveProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Change a supported property such as text, optional or hidden on one objective.

Field / stored key	What to enter	Starting value
Quest `questId`	Choose a Quest Designer ID.	Blank
Stage `stageId`	Choose a stage in that quest.	Blank
Objective `objectiveId`	Choose an objective in that stage.	Blank
Objective Property `property`	Choose an objective property.	Blank
New Value `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Reveal an optional objective after the player reads a note. Choose the correct quest and stage before the objective. Completing an objective remains a separate operation.

Saving, language and project settings

Runtime settings and save files are different kinds of persistence. Test saves with a fresh game, and use the same slot numbering consistently in scripts and custom save widgets.

Save Game

ID: SaveGame | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Save through the basic runtime route. With no structured slot supplied, this route uses slot 0.

No fields in the standard catalog form.

In use. Use it only when a deliberate quick-save slot is wanted. The normal catalog form has no fields. Choose Save Game Slot when the player or script must select a slot and a name.

Load Game

ID: LoadGame | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Load through the basic runtime route, defaulting to slot 0.

No fields in the standard catalog form.

Timing: Awaits loading.

In use. Use Load Game Slot for an explicit slot or a custom load screen. Loading replaces important live state; keep later actions in the old sequence to a minimum and test missing-file behavior.

Save Game Slot

ID: SaveGameSlot | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Save the current game into a chosen slot with an optional name.

Field / stored key	What to enter	Starting value
Slot `slot`	Explicit save slot number. Selected-slot widget commands use widget selection instead.	0
Save Name (optional) `saveName`	Optional player-facing name for the saved game. May be blank.	Blank

In use. Save slot 1 as "Before the archive." Save a coherent completed puzzle state, rather than halfway through a multi-step reward. Do not silently overwrite a player-chosen slot without a sensible interface policy.

Load Game Slot

ID: `LoadGameSlot` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Load the chosen save slot.

Field / stored key	What to enter	Starting value
Slot `slot`	Explicit save slot number. Selected-slot widget commands use widget selection instead.	0

Timing: Awaits loading.

In use. Load the row the player selected in a custom save screen. Test an empty slot and a save from another room. OnSaveLoaded is the hook for follow-up work that belongs to successful loading.

Set Save Slot Image

ID: `SetSaveSlotImage` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Set the image override used for save-slot presentation through the extension.

Field / stored key	What to enter	Starting value
Save Slot Image `assetId`	Choose an imported image.	Blank

In use. Choose an authored chapter illustration instead of a live screenshot when appropriate. The asset must be an imported image. It does not create a new save or choose which slot gets saved.

Set Game Language

ID: `SetGameLanguage` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set the game text language, optionally persisting the selection.

Field / stored key	What to enter	Starting value
Language Code `language`	Language code used by the project, such as en.	Blank
Save Player Preference `persist`	Remember this language preference beyond the immediate runtime change.	true

In use. Choose a language code present in the project's translations. Keep Persist on for a player preference that should survive later sessions. This does not translate text that was never entered into the translation workflow.

Change Voice Language

ID: `ChangeVoiceLanguage` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set the runtime voice language.

Field / stored key	What to enter	Starting value
Language `language`	Language code used by the project, such as en.	Blank

In use. Choose the voice-language code that has the corresponding audio resources. It is separate from subtitle language, allowing one voice language with another subtitle language.

Change Subtitle Language

ID: `ChangeSubtitleLanguage` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Set the runtime subtitle language.

Field / stored key	What to enter	Starting value
Language `language`	Language code used by the project, such as en.	Blank

In use. Change subtitles while leaving voice in the original language. Test custom GUI text and dynamically created lines too; fixed strings outside the translation system may not change automatically.

Change Project Property

ID: `ChangeProjectProperty` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Assign a supported project property through a property path.

Field / stored key	What to enter	Starting value
Property `property`	Choose a project property path.	Blank
To `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Change a runtime project option offered by the picker. Match the value type and test whether the relevant manager reads it immediately. This is not an instruction to rewrite the saved editor project file.

Change Game Setting

ID: `ChangeGameSetting` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

Assign a supported runtime game setting.

Field / stored key	What to enter	Starting value
Setting `setting`	Choose a runtime game setting.	Blank
To `value`	A value of the type required by the selected variable, operation or property.	Blank

In use. Change a control or presentation setting used by a custom options screen. Some systems apply settings when a scene or manager is rebuilt, so confirm the live result instead of assuming every property is reactive.

Cutscene control and timeline-specific actions

A timeline event has a start time and an action. Its event duration and the action's own timing are not always interchangeable. These five legacy pose/movement entries are offered by the Cutscene Timeline; they are not regular shared-list picker entries.

Cutscene: Start

ID: `StartCutscene` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper · Timeline

Start a Cutscene Timeline record and wait for its session.

Field / stored key	What to enter	Starting value
Cutscene `cutsceneId`	Choose a Cutscene Timeline ID.	Blank

Timing: Waits for the cutscene session.

In use. Choose Reveal the Signal from the lever's successful branch. Let the cutscene's input, HUD, skip and camera-restoration settings own the cinematic behavior. Avoid starting the same cutscene from its own first event.

Cutscene: End

ID: `EndCutscene` | Native inspector · Shared list / GUI · Visual graph · Text action wrapper

End the active cutscene, optionally restoring the camera.

Field / stored key	What to enter	Starting value
Restore Previous Camera `restoreCamera`	Restore the cutscene's previous camera when ending.	true

In use. Use it when a branch should cut the sequence short. Keep critical puzzle state outside optional visual beats so skipping and ending early cannot leave the game half-repaired.

Set Camera

ID: `SetCamera` | Timeline

Set an explicit camera position, look-at target and optional field of view. This is a timeline/raw-action route.

Field / stored key	What to enter	Starting value
Position `position`	Destination/placement coordinates as [X, Y, Z].	[0, 0, 0]
Look-at Target `target`	[X, Y, Z] numeric array.	[0, 0, -1]
FOV (negative keeps current) `fov`	Number; decimals allowed.	-1

The fields above describe the timeline/raw action data. They are not a shared-list form.

In use. Use [0,3,7] looking at [0,1,0] for a staged view. The target is a world point, not Euler rotation. Prefer an authored camera plus Activate Camera when you want an easily reusable view.

Camera Move

ID: `CameraMove` | Timeline

Move the camera using the pose and timing contained in the timeline action data.

Field / stored key	What to enter	Starting value
End Position `position`	Destination/placement coordinates as [X, Y, Z].	-
End Look-at Target `target`	[X, Y, Z] numeric array.	-
Duration (seconds) `duration`	Time in seconds for this action. Read its timing note before assuming it waits.	-
FOV (optional) `fov`	Number; decimals allowed.	-

The fields above describe the timeline/raw action data. They are not a shared-list form.

Timing: Waits for the movement routine.

In use. Use a slow push toward the signal lamp. Check Position, Target and Duration together. A following/tracking camera can compete with manual motion; establish which camera system owns the view during the shot.

Walk To

ID: `WalkTo` | Timeline

Use the legacy timeline walk action with an entity ID, position and speed.

Field / stored key	What to enter	Starting value
Entity `entityId`	Choose a scene entity ID.	-
Position `position`	Destination/placement coordinates as [X, Y, Z].	-
Speed `speed`	Movement speed; 0 uses the normal movement speed where supported.	-

The fields above describe the timeline/raw action data. They are not a shared-list form.

In use. Use it for an existing timeline that already contains staged walks. For new shared-list movement, Character Walk To provides the newer character/target route. Neither action removes physical obstacles along the route.

Set Character State

ID: `SetCharacterState` | Timeline

Set a character's named animation state through the legacy timeline route.

Field / stored key	What to enter	Starting value
Character Entity `entityId`	Choose a scene entity ID.	-
State ID `stateId`	Text.	-

The fields above describe the timeline/raw action data. They are not a shared-list form.

In use. Set Elias to a configured listening or idle state. A state ID belongs to the Character Designer's mapping; it is not necessarily the imported animation clip's name.

Toggle Light

ID: `ToggleLight` | Timeline

Toggle an authored light entity, optionally mirroring the new state in a variable.

Field / stored key	What to enter	Starting value
Light Entity `entityId`	Choose a scene entity ID.	-
State Variable (optional) `variable`	Text.	-

The fields above describe the timeline/raw action data. They are not a shared-list form.

In use. Switch a practical lamp light during a timeline. It controls the light entity rather than a separate emissive mesh or screen bloom. If the visible bulb must glow too, control that object's glow separately.

Local world events: quick reference

Objects, triggers and interactive regions use local responses. In scene entity records, volume/pointer actions live in the trigger event data; adventure verbs live in hotspot.verbs. The inspector presents those as separate tabs.

Stored local key	Visible meaning	Typical use
onEnter	Player Collision / Entry	Start a guarded greeting when the active player enters.
onExit	Player Exit	Remove a temporary local effect after walking out.
onClick	Left click	Run a special direct pointer response.
onDoubleClick	Left double click	A deliberate alternate pointer response; account for the first click.
onRightClick	Right click	Inspect or cancel, according to your control design.
onDoubleRightClick	Right double click	An optional alternate right-pointer response.
onHover	Pointer enters/hovers	Lightweight presentation or a short contextual update.
onHoverExit	Pointer leaves	Clear a temporary hover presentation.

Legacy onCollide data can supply entry work when the newer onEnter list is empty. For new work, use the named entry/exit lists shown by the inspector rather than building two overlapping versions of the same collision response.

The runtime guards overlapping execution of an entity's same local event while it is running. Later events can still run again. Important one-time behavior needs a completion flag that is part of the saved game state.

A visible region effect and an interactive region are not automatically the same thing. Enable/configure the region's interaction behavior and overlap/click setup before expecting its events to run. These volume events are not a general notification for every physical object collision.

GUI mouse events: quick reference

Event key	Meaning	Good use
hover	Pointer enters the element's hover route.	A light contextual update; use hover images/styles for appearance.
click	Left/touch click route.	A button command, primary behavior and additional list.
doubleClick	Left double-click route.	An alternate activation where first-click behavior is harmless.
rightClick	Right-click route.	An inspection or optional secondary action.
doubleRightClick	Right double-click route.	An optional secondary activation.

Common GUI pointer context includes guiId, guiElementId, guiEvent and eventName such as GUI.click. Those are context values for the selected callback, not extra names to check in the shared script subscription panel.

The primary event work runs before its additional actions. A selected event Verb can take the verb-selection path instead of the primary behavior. The additional list is still considered afterward. Keep only one clear intention in each part of the event.

> There is no common GUI mouse-release action-list key in this five-event set. Returning a pressed image to normal/hover after release is visual state behavior, not a reason to run a second gameplay reward.

Save widget and progress callback data

A save widget forwards its event with additional slot information. Use saveExists before assuming a row has a file, a thumbnail or a meaningful scene name.

Context key(s)	Meaning
saveSlot	The slot number selected by this event.
saveExists	Whether the slot contains a save.
saveName	The save's user-facing name.
saveSceneId, saveSceneName	The scene recorded by the save.
saveTimestamp, saveDate, saveTime	Timestamp and formatted date/time values.
savePlayTime, savePlayTimeSeconds	Display and numeric playtime values.
saveThumbnailPath	Thumbnail path where available.
saveIsAutosave	Whether this row represents an autosave.
saveStatus	saved or empty.
guiElementId	The widget element that raised the event.

A progress threshold callback supplies guiElementId, percentage and value. Its configured script and action work belong to that threshold, not to an invented shared OnProgress event. If a progress bar can move backward and cross a threshold again, check its repeat policy and guard any one-time reward.

A returned slot number is data. Decide explicitly whether the response should select, load, save, rename or delete it. Browsing a row should not accidentally overwrite or remove a save.

Advanced: structured action data

Most work should use the visible forms. Structured action data is useful when reviewing exported records, calling helpers from scripts or working around a field that a particular form does not expose. An action object contains type plus that action's stored field keys.

```
{
  "type": "ShowMessage",
  "text": "The signal is on.",
  "duration": 2.2,
  "blockInput": false
}
```

A list is a JSON array of those objects. Its order is the order of execution. Valid JSON uses double quotes, true/false without quotes for Booleans, and no trailing comma.

```
[
  {"type":"ChangeVariable", "name":"powered",
   "operation":"set", "value":true},
  {"type":"PlaySound", "assetId":"chime",
   "channel":"puzzle"},
  {"type":"ShowMessage", "text":"The signal is on.",
   "duration":2.2}
]
```

Paste a value into the correct field or record location. A list array is not a whole scene record, and a cutscene's event object wraps its action inside an action member. Keep a project backup before editing large raw records.

Advanced: a nested conditional in raw data

The runtime supports an IfVariable action with nested then and else arrays. Existing project data can contain this form. It is not a condition row offered by the normal shared sequence picker. A visual graph is easier to edit and inspect for most authors.

```
{
  "type":"IfVariable",
  "name":"key_taken",
  "equals":true,
  "then":[
    {"type":"ShowMessage", "text":"I have the key.",
     "duration":2.2}
  ],
  "else":[
    {"type":"ShowMessage", "text":"I need the key.",
     "duration":2.2}
  ]
}
```

Only the selected nested branch runs. After it finishes, the outer list continues with its next action. Do not confuse that branch selection with a returned false value from some other action; the ordinary list runner does not interpret every result as a condition.

The example was checked in the runtime. For a new authoring workflow, use If Variable, Has Inventory or If Compare in a graph, then call the graph from the inspector.

A story flag such as key_taken answers whether the pickup happened. Has Inventory answers whether the character currently carries the item. Choose the condition that matches the rule; those can differ after an item is consumed or transferred.

Version notes for these lessons

This book documents PCA 3D Adventure Studio 0.86.3.4. The engine was inspected and the examples were exercised in its native editor/runtime. These points matter when your screen does not match the idea of a single identical action form everywhere.

Current detail	Practical route
Native picker has fewer entries than the shared sequence picker.	Use Change Variable / Set for assignment. Call a visual/text script for commands missing from that inspector.
Some shared Change Scene forms omit sceneId.	Use the native destination controls or the visual Change Scene node; call that graph from GUI events.
New GUI catalog primary actions and changed additional row types lose the stored type in this build.	Use the convenience run-visual-script behavior and keep the sequence in that script. Existing valid structured catalog actions and lists execute at runtime.
The visual On Event picker lists 15 names; the text subscription panel lists 18.	Use dedicated video/music graph entries and the appropriate script-ready route.
Play Music exposes Skippable but the action bridge does not use it.	Implement deliberate stopping/continuation logic instead of relying on that field.
Most imported animation requests return after starting.	Use explicit Wait/timeline scheduling when later work must be delayed.
Duration compatibility treats values over 50 as milliseconds for legacy speech/messages.	Use short current second values and a proper timeline for long pacing.
Generic property edits do not rebuild every live resource.	Use dedicated visibility, motion, glow and media actions where available.

The documentation does not change the engine or any user project. If a later build fixes one of these limitations, use its improved editor controls while keeping the event ordering and state-management principles in these lessons.

Terms you will see in the editor

Term	Plain meaning
Action	One instruction to the running game.
Action list / sequence	Several instructions executed in their stored order.
Event	A notification or callback that starts work. A timeline event is a scheduled item instead.
Context	Information supplied for this particular call.
Caller / this	The game record captured as the owner of this call, if one exists.
ID	A stable stored reference, distinct from a display name.
Definition	An authored record used as a source, such as an item or character.
Instance	A live copy of a definition in the running scene.
Variable	A named value used to remember game state.
Boolean	An on/off value: true or false.
Variant / JSON value	A field that can hold different types, such as a number, string, array or dictionary.
Condition / branch	A test that selects which path runs.
Subscription	A script's request to listen for a named shared event.
Global instance	A script instance kept alive between calls.
Await / Wait	A caller pausing until the called routine returns or a deliberate delay finishes.
Guard	A condition/flag that prevents unwanted repeated work.
Modal / input lock	A state in which a UI or sequence owns input and normal play is restricted.
Spawn	An authored arrival position in a room.
Runtime	The game while it is running, including the embedded preview.

The most useful distinction is simple: state says what is true, presentation shows it, and an event tells the game when to do some work. Keep those jobs clear and complicated puzzles become much easier to maintain.

Alphabetical command index

Each name is a link to its reference entry. The stored ID is included underneath for searching scripts and structured action data. Availability and specialized fields are explained at the destination.

Activate / Blend to Camera
ActivateCamera

Add Asset to Scene
AddAssetToScene

Add Character To Scene
AddCharacterToScene

Add Dialogue Choice
AddDialogueChoice

Add Dialogue Choice Node
AddDialogueChoiceNode

Add Dialogue Line
AddDialogueLine

Add Inventory
AddInventory

Add Score
AddScore

Align Attach Points
AlignAttachPoints

Animation: Pause
PauseAnimation

Animation: Resume
ResumeAnimation

Attach Object
AttachObject

Attach Particle Effect
AttachParticleEffect

Attach Points
AttachPoints

Attach Points Touching
AttachPointsTouching

Automatic Input Switching
SetAutoInputSwitching

Call Script Function
CallScriptFunction

Camera Move
CameraMove

Camera Orbit
CameraOrbit

Camera Shake
CameraShake

Cancel Two-Object Command
CancelPreposition

Carousel: Add Image
CarouselAddImage

Carousel: Clear Images
CarouselClearImages

Carousel: Next Image
CarouselNext

Carousel: Previous Image
CarouselPrevious

Carousel: Remove Image
CarouselRemoveImage

Center GUI Model
CenterGUIModel

Change Camera
ChangeCamera

Change Character Description
ChangeCharacterDescription

Change Character Property
ChangeCharacterProperty

Change Current Verb Command
ChangeVerb

Change Game Cursor
ChangeGameCursor

Change Game Setting
ChangeGameSetting

Change Gui Element Property
ChangeGUIElementProperty

Change Gui Property
ChangeGUIProperty

Change Inventory Description
ChangeInventoryDescription

Change Inventory from Player to Player
TransferInventory

Change Inventory Quantity
ChangeInventoryQuantity

Change Object Description
ChangeObjectDescription

Change Object Property
ChangeObjectProperty

Change Project Property
ChangeProjectProperty

Change Region Description
ChangeRegionDescription

Change Room Camera Bounds
SetRoomCameraBounds

Change Scene
ChangeScene

Change Scene Description
ChangeSceneDescription

Change Subtitle Language
ChangeSubtitleLanguage

Change Variable
ChangeVariable

Change Voice Language
ChangeVoiceLanguage

Character Instant Teleport To
CharacterTeleport

Character Say
CharacterSay

Character Walk Through Waypoints
CharacterWalkPath

Character Walk To
CharacterWalkTo

Character Waypoints: Add Waypoint
AddCharacterWaypoint

Character Waypoints: Clear Queue
ClearCharacterWaypoints

Character Waypoints: Walk Queue
CharacterWalkWaypoints

Close Interaction Coin
CloseInteractionCoin

Create Primitive
CreatePrimitive

Create Procedural
CreateProcedural

Create Variable
CreateVariable

Cutscene: End
EndCutscene

Cutscene: Start
StartCutscene

Cycle Verb
CycleInteractionVerb

Despawn Scene Object
DespawnObject

Detach Object
DetachObject

Detach Particle Effect
DetachParticleEffect

Disable Gui
DisableGUI

Disable Gui Element
DisableGUIElement

Enable / Disable Automatic Interaction Approach
SetInteractionApproachEnabled

Enable / Disable Character Action
SetCharacterActionEnabled

Enable / Disable Dialogue Choice
SetDialogueChoiceEnabled

Enable / Disable Flashlight Mode
SetFlashlightMode

Enable / Disable Object
SetObjectEnabled

Enable / Disable Object Action
SetObjectActionEnabled

Enable / Disable Player Input
SetPlayerInputEnabled

Enable / Disable Trigger
SetTriggerEnabled

Enable / Disable Verb Command Usage
SetVerbUsageEnabled

Enable Gui
EnableGUI

Enable Gui Element
EnableGUIElement

Enable Lip Sync
SetLipSyncEnabled

Enable Mouse Clicks
SetMouseClicksEnabled

Enable Mouse Pointing
SetMousePointingEnabled

End Dialogue
EndDialogue

End Visual Script
EndVisualScript

Execute Selected Verb Command
ExecuteVerb

Execute Two-Object Command
ExecutePreposition

Face Character
FaceCharacter

Face Coordinates
FaceCoordinates

Face Object
FaceObject

Follow Character
FollowCharacter

Get Attach Point
GetAttachPoint

Get Attach Point Contacts
GetAttachPointContacts

Get Control State
GetControlState

Give Inventory
GiveInventory

Hide / Unhide Object
SetObjectVisible

Hide Gui
HideGUI

Hide Gui Element
HideGUIElement

Hide Player Character
HidePlayerCharacter

Interact with Nearest Hotspot
InteractNearest

Is Current Room
IsCurrentRoom

Jump
Jump

List Box: Add Item
ListBoxAddItem

List Box: Clear Items
ListBoxClearItems

List Box: Delete Item
ListBoxDeleteItem

List Box: Modify Item
ListBoxModifyItem

Load Game
LoadGame

Load Game Slot
LoadGameSlot

Melt Accumulated Snow
MeltSnow

Move NPC to a Different Room
MoveNPCToRoom

Music: Fade
FadeMusic

Music: Pause
PauseMusic

Music: Resume
ResumeMusic

Open Interaction Coin
OpenInteractionCoin

Pause FMV
PauseFMV

Pause Voice
PauseVoice

Play Animation
PlayAnimation

Play Gameplay Recording
PlayGameplayRecording

Play GUI Model Animation
PlayGUIModelAnimation

Play Music
PlayMusic

Play Sound
PlaySound

Play Video
PlayVideo

Play Voice
PlayVoice

Player Say
PlayerSay

Proximity Interaction
SetProximityInteractionEnabled

Quest: Change Objective Property
ChangeQuestObjectiveProperty

Quest: Change Property
ChangeQuestProperty

Quest: Change Stage Property
ChangeQuestStageProperty

Quest: Complete Objective
CompleteObjective

Quest: Complete Quest
CompleteQuest

Quest: Fail Objective
FailObjective

Quest: Fail Quest
FailQuest

Quest: Hide / Unhide
SetQuestHidden

Quest: Reset Objective
ResetObjective

Quest: Reset Quest
ResetQuest

Quest: Set Stage
SetQuestStage

Quest: Track / Untrack
SetQuestTracked

Remove Attach Point
RemoveAttachPoint

Remove Dialogue Choice
RemoveDialogueChoice

Remove Dialogue Node
RemoveDialogueNode

Remove Inventory
RemoveInventory

Reset Dialogue History
ResetDialogueHistory

Resume FMV
ResumeFMV

Resume Voice
ResumeVoice

Run Visual Script
RunVisualScript

Save Game
SaveGame

Save Game Slot
SaveGameSlot

Save Game Widget: Command
SaveGameWidgetCommand

Say
Say

Screen Fade
FadeScreen

Screen Transition
ScreenTransition

Seek FMV
SeekFMV

Seek Video
SeekVideo

Seek Voice
SeekVoice

Set 2D Camera
Set2DCamera

Set 2D Entity Depth
Set2DEntityDepth

Set 2D Layer Parallax
Set2DParallax

Set Advanced Camera Behavior
SetCameraBehavior

Set Attach Point
SetAttachPoint

Set Camera
SetCamera

Set Camera Priority
SetCameraPriority

Set Camera Targets
SetCameraTargets

Set Character State
SetCharacterState

Set Control Setting
SetControlSetting

Set Game Language
SetGameLanguage

Set Glow
SetGlow

Set Input Device
SetInputDevice

Set Interaction Mode
SetInteractionMode

Set Motion
SetMotion

Set Movement Profile
SetMovementProfile

Set Object Interaction Approach
SetObjectInteractionApproach

Set Retro Rendering Style
SetRetroStyle

Set Room Brightness
SetSceneBrightness

Set Room Dimension
SetRoomDimension

Set Room Tint
SetSceneTint

Set Runtime Controls Enabled
SetRuntimeControlsEnabled

Set Save Slot Image
SetSaveSlotImage

Set Snow Accumulation
SetSnowAccumulation

Set Variable
SetVariable

Set Voice Speaker
SetVoiceSpeaker

Shake Screen
ShakeScreen

Show / Hide 2D Layer
Set2DLayerVisible

Show Gui
ShowGUI

Show Gui Element
ShowGUIElement

Show Message
ShowMessage

Show Mouse Pointer
SetMousePointerVisible

Skip Ahead in Soundtrack
SeekMusic

Sound: Fade
FadeSound

Sound: Pause
PauseSound

Sound: Resume
ResumeSound

Spawn Scene Object
SpawnObject

Start Dialogue
StartDialogue

Start FMV
StartFMV

Start Gameplay Recording
StartGameplayRecording

Start Quest
StartQuest

Stop Animation
StopAnimation

Stop Attach Point Watch
StopAttachPointWatch

Stop Camera Orbit
StopCameraOrbit

Stop FMV
StopFMV

Stop Follow Character
StopFollowCharacter

Stop Gameplay Playback
StopGameplayPlayback

Stop Gameplay Recording
StopGameplayRecording

Stop Music
StopMusic

Stop Screen Shake
StopScreenShake

Stop Sound
StopSound

Stop Startup Splash Audio
StopStartupSplashAudio

Stop Video
StopVideo

Stop Voice
StopVoice

Stop Zoom Blur
StopZoomBlur

Switch Character
SwitchCharacter

Toggle Glow
ToggleGlow

Toggle Light
ToggleLight

Toggle Motion
ToggleMotion

Toggle Rgb String
ToggleRGBString

Unlock Hint Topic / Answer
UnlockHint

Walk To
WalkTo

Watch Attach Point Contacts
WatchAttachPointContacts

Zoom Blur
ZoomBlur

Reference basis and lesson checks

Prepared for PCA 3D Adventure Studio 0.86.3.4. Command availability and common fields were checked against the build's action catalog and native designers. Event names, standard payloads, caller behavior and execution order were checked against its runtime dispatch routes.

The walkthrough used the actual native editor and embedded runtime, with bundled assets where useful. Captures show the real controls and gameplay presentation; they are not mockups of a proposed interface. Examples remain deliberately small so they can be reproduced without a separate practice download.

Check performed	Result relevant to the guide
Ordered value changes	Later actions saw the earlier modified value.
Create Variable called twice	The existing value was retained.
Failed animation request followed by assignment	The later list instruction still executed.
Raw IfVariable branch	Only its selected nested branch applied its test result.
Custom-event broadcast	A subscribed text script and On Event graph both received it.
this with and without an object caller	Local targeting worked; ownerless targeting was skipped.
One-time greeting graph called twice	The completion flag stayed set and prevented repeated presentation.
GUI primary plus valid additional actions	The primary state change and additional presentation executed.
Key pickup	Inventory gained the key, score changed and the world pickup was hidden.
Door without/with the key	The locked response stayed in the room; the valid branch entered the archive.
Room return	Entry count increased without repeating first-entry state.

The included engine documentation consulted for terminology and authoring routes includes Functions and Caller Context, Native Scripting, Visual Scripting, Dialogue Designer, Cutscene Timeline, Screen Shaders and GUI Events, Save Game Widget, and Runtime Control Profiles. Where older documentation differs from the checked build, this guide describes the checked behavior.