

RPG Action Bar Kit

Complete Buyer Guide

Version 1.1.0 | Unreal Engine 5.8 | Windows

Contents

Start here: get your first bar

Included content and dependencies

Connect one ability: simple Blueprint route

Connect an existing ability system: provider route

Supply icons, cooldowns and live availability

Use the bar and manage player setups

Troubleshoot an integration

Customize appearance, keys and your own actions

Profiles, persistence and cleanup

Native integration and gameplay ownership

Optional Auto Attack showcase

Edit equipment-category icons

Editable Blueprint integration walkthroughs

Version 1.1.0 buyer guide. Unreal Engine 5.8 on Win64.

Start here: get your first bar

1. Install the engine-compatible plugin through Fab. For a manual source installation, put ActionBarKit in your existing project's Plugins directory and build Development Editor with Unreal's compatible Windows C++ toolchain.
2. Enable BIS RPG Action Bar in Edit > Plugins. Keep the engine ACLPlugin dependency enabled for the compressed punch clips. Restart if Unreal requests it. Enable Show Plugin Content in the Content Browser.
3. Press Play with a possessed local player. The automatic host displays the bar. If the starter actions have not been assigned, open the gear panel > Preview > Load Action Bar to assign its twelve demonstration actions.
4. Click a slot or use its displayed key. The standard first row is 1-9, 0, minus and equals. Preview buttons show cooldown feedback; the optional local Auto Attack example supplies the punch/melee demonstration. Your game still owns its actual abilities and resources.
5. If the bar is missing, confirm a local PlayerController exists and automatic startup is enabled in Project Settings > Plugins > BIS RPG Action Bar. Start Action Bar Kit can also reuse the existing host explicitly; do not create a second displaying host.

Included content and dependencies

1. This buyer guide covers descriptor version 1.1.0 for Unreal Engine 5.8 on Win64. The plugin contains 32 Unreal assets and two runtime modules: ActionBarKit and BISActionBarBeginnerAbilities. Manual source installation needs Unreal's supported Windows C++ compiler.
2. The optional example includes three unarmed attack clips (two punches and a roundhouse kick) and eight BIS equipment-category icons. Other action buttons remain a showcase for cooldown and charge presentation. Icons do not add sword attacks, bow firing, spells or an inventory.
3. ACLPlugin is the required engine-distributed animation dependency. The standalone project's mannequin, environment and ABP_Unarmed are demonstration-host content, not part of the plugin download. GAS, multiplayer prediction and production animation-notify hit timing are not supplied.
4. Auto Attack toggles on and off, including off during cooldown. Walking/running use punches only with locomotion legs and steering. Standing can use the full three-clip cycle, including the roundhouse. Starting movement fades an owned kick; stopping makes the kick eligible for the next attack. Optional character setup is in the equipment/showcase reference; ordinary action-bar use does not require it.

Connect one ability: simple Blueprint route

Use this route when you want to forward slot requests into an existing Blueprint without implementing the provider interface. The default starter IDs are Action01 through Action12; Fireball is a buyer-authored example.

1. After your local PlayerController exists, call Start Action Bar Kit from BeginPlay or your UI initialization event. Promote the returned Action Bar Host to a retained variable and check it is valid. Repeated startup calls reuse the existing host.
2. Create an ABKActionBarPreset with one slot: SlotIndex 1, stable ActionId Fireball, DisplayName Fireball, your imported Texture2D in Icon, and your chosen bound key. Connect the preset at first startup, or use Set Slots to supply the slot data. Set Simple Action Bar Slot changes an ID and label; it does not supply an icon.
3. Bind On Action Slot Requested on the retained host. Break the Action Slot Request and Switch on Name using ActionId. In the Fireball case, ask your existing ability owner to cast. Clicking the slot and pressing its key reach this same hook.
4. For a standalone bar, call Start Slot Cooldown with SlotIndex 1 and Duration Seconds 5 only after your ability owner accepts the cast. The bar displays and counts down the cooldown and blocks that slot. Clear Slot Cooldown ends the display lock early. Your ability owner still enforces its own gameplay cooldown; use the provider route for live authoritative state.
5. Use Set Slot Availability With Reason for an unavailable standalone action and its tooltip explanation, such as Requires 20 mana. Update availability when your gameplay state changes. Do not run the ability again from On Action Activated.
6. During teardown remove only your own event receiver. If you reconnect, remove the exact receiver before binding it again so one request reaches your owner once.

Connect an existing ability system: provider route

1. Choose this route when your ability, combat or inventory system owns live slot assignments, resource checks and cooldowns. Implement ABKActionProviderInterface on your owned object, or adapt BP_ABK_BuyerProvider in your project.
2. Connect Buyer Action Bar to the existing host in the editable actor example, or call Bind Action Provider with your interface object. Branch on the returned Boolean and Failure. A rejected connection must preserve the current owner.
3. Get Action Bar Slots returns your current snapshot. Can Trigger Action Slot checks readiness without effects. Trigger Action Slot performs the final check, commits your existing ability once, and returns true only when accepted.
4. Provider-backed activation calls Trigger Action Slot; it does not broadcast the standalone On Action Slot Requested event. On Action Activated observes an accepted provider request. In the simple event route it reports that the bar processed the request, not proof that your ability owner accepted it. Use your owner's result for cast success, and keep this observer free of ability execution.
5. Handle move, clear, plain-key and complete-chord requests in your provider. An accepted mutation is followed by a refreshed snapshot; reject an operation your game does not allow. Do not apply the same swap or clear a second time in the host.
6. Disconnect before replacing the owner or changing hosts. Remove only your exact receivers. The supplied disconnect example also cleans up at EndPlay.

Supply icons, cooldowns and live availability

1. Keep SlotIndex positive and unique and ActionId stable. Supply DisplayName, Description, Icon, bound key/chord and count data through FABKActionSlotViewData. Changing a label or icon must not change the saved action identity.
2. For Fireball with a five-second cooldown, return CooldownDuration 5 and CooldownRemaining equal to your gameplay owner's current remaining time, for example 3.2 seconds. Return zero remaining time when ready; keep the original duration while it is cooling down.

3. Call Refresh From Action Provider after external cooldown, resource, availability, count or icon changes, and on your chosen update cadence for a live countdown. The host reads your new snapshot; it does not advance the provider's gameplay cooldown for you.
4. Supply enabled/usable state, Availability and UnavailableReason from your owner. A visible countdown is presentation; Can Trigger Action Slot and Trigger Action Slot must still enforce the real cooldown and resource rules.
5. Start Slot Cooldown and Clear Slot Cooldown are for standalone bars and are refused on provider-backed bars. Keep one cooldown authority. Reference custom textures/presets in project assets so they are included when your game is cooked.

Use the bar and manage player setups

1. Activate an assigned slot by clicking it when your game exposes the cursor, or press its displayed keyboard/gamepad binding. The bar needs no separate Enhanced Input mapping for its own key polling. Your game retains control of cursor and input mode.
2. In a standalone setup, open the gear panel > Preview. Load Action Bar assigns the twelve demonstration actions while preserving keys; it replaces preview assignments and is disabled when a provider owns the bar. Drag a sample onto a slot to assign it, drag between slots to swap, or drop an assignment outside the bar to clear it.
3. Action Slots Locked blocks assigning, swapping and drag-off clearing while leaving activation available. Right-click clears a keybind only while Key Binds mode is active; it does not remove the assigned action.
4. Input > Key Binds: hover a slot, hold any supported modifiers and press the desired keyboard/gamepad button. Backspace/Delete or right-click clears that hovered binding. Escape clears the hovered or explicitly captured slot; with no target it exits Key Binds mode.
5. Use Profiles to create, load, rename, save or delete a named player setup and select its default. The last profile cannot be deleted. Existing player saves take precedence over a new project preset. Provider-owned gameplay assignments and live cooldowns remain in your own save system.
6. Call Show Action Bar or Hide Action Bar when your game controls HUD visibility. Use the gear panel for appearance, bar groups, labels and cooldown presentation; movement locks and category resets preserve unrelated settings.

Troubleshoot an integration

1. No bar or an invalid host: initialize after a local PlayerController exists and check automatic startup in Project Settings > Plugins > BIS RPG Action Bar. Dedicated servers do not display the local UI.
2. Two bars: reuse Start Action Bar Kit and avoid adding another auto-displaying host to the same controller.
3. Clicks do nothing: check your cursor/input mode and bind the hook for your chosen route. Standalone uses On Action Slot Requested; a provider uses Trigger Action Slot.
4. Cooldown or icon is stale: update the provider snapshot and call Refresh From Action Provider. Check the texture reference and cooking inclusion. Do not use standalone cooldown helpers with a provider.
5. An ability fires twice: commit it in one request handler and keep On Action Activated observational. Remove your exact receiver before re-binding.
6. Your new preset seems ignored: inspect the existing player profile first. Do not delete or reset a player's saved setup just to apply a development default.

Customize appearance, keys and your own actions

1. Open the gear panel. General changes overall scale, glass, border and movement. Bars selects all bars or one numbered group; Bar appearance adjusts slot size, artwork, spacing and opacity. Labels and Cooldowns control presentation.
2. Use Input > Key Binds, hover a slot and press the desired keyboard/gamepad button with any supported modifiers. Right-click or Backspace/Delete clears only that binding while rebinding. Escape clears the hovered binding, or leaves capture mode when no slot is hovered.
3. For a starting layout, create an ABKActionBarPreset Data Asset, edit Layout and Slots, and select it as Default Preset in plugin Project Settings. Existing saved settings take precedence; do not reset a player's setup merely to change the project default.
4. Give each buyer-defined action a stable ActionId and update its label, icon, enabled/usable state and cooldown data. Use Set Slots or your provider snapshot. Changing artwork must not change the saved action identity.
5. For your game's effects, connect the editable buyer provider below. Keep readiness read-only, commit the effect once in Trigger Action Slot, return acceptance and refresh state. Do not apply gameplay again from the notification callback.

Profiles, persistence and cleanup

1. Use the provider's Save Buyer Presentation and Reload Buyer Presentation functions for UI setup. Named profile functions create, load, rename, delete and choose a default profile. Keep destructive confirmation separate: request, then confirm or cancel.
2. UI persistence stores presentation and standalone configuration under the host's SaveGameSlotName/UserIndex. It does not save or recreate your gameplay inventory, combat state or provider authority. Your existing game owns those saves.
3. For slot removal, use Request Buyer Clear Confirmation, then Confirm Buyer Clear or Cancel Buyer Clear Confirmation. Cancelling preserves the assignment. Reset/profile deletion have their own request, confirm and cancel paths.
4. Disconnect this provider before changing hosts or during teardown. Remove only this provider's receivers. Rebinding must not duplicate callbacks or clear another UI owner. EndPlay also cleans up the example's connection.
5. For a packaged game, reference custom icons, presets and textures so they are cooked. A soft path stored only in player SaveGame data is insufficient proof of inclusion. Inspect the packaged result separately.

Native integration and gameplay ownership

1. The equivalent public C++ recipe is bundled in Reference/NativeBuyerIntegration.md. Its minimal provider lives in Source/ActionBarKit/Public/ABKDemoActionProvider.h and the corresponding Private .cpp. Add ActionBarKit to your existing game module's dependencies; buyer Blueprint use does not require modifying plugin source.

2. Implement `IABKActionProviderInterface`'s seven callbacks: slot snapshot, readiness, trigger, move, clear, bound key and complete chord. `BindActionProvider` returns acceptance plus Failure. `RequestTriggerSlot` returns the provider's acceptance; `OnActionActivated` observes an already accepted request.
3. `OnBagRequested` asks your inventory owner to open its window. `RecordDamageForDps` receives damage already confirmed by your combat owner. Neither callback supplies production inventory transactions or applies damage a second time.
4. The optional beginner native route adds `BISActionBarBeginnerAbilities`. `RequestAutoAttack` returns acceptance, hit/miss, target, applied damage, reason and punch index. Initialize after local possession; remove your receiver and call `ShutdownBeginnerAbilityBar` during cleanup.
5. Check normal requests, invalid references, rejection, cooldown, cancellation, disconnect/rebind and save/reload in your own host. The release candidate's native and buyer Blueprint workflows have been validated in the existing validation host. Packaged validation remains open for this local draft.

Optional Auto Attack showcase

1. The optional local example toggles Auto Attack on and off, including off during cooldown. Standing uses two punches and a roundhouse; walking/running selects punches while preserving locomotion and steering.
2. Use `ExecuteAutoAttack` only when adapting this optional combat example. Your existing combat owner still decides effects, targeting, resources and damage. Equipment-category icons do not implement weapon combat.
3. The download does not include the standalone character or its Animation Blueprint. If you want to reproduce that demonstration on a compatible character, follow `Reference/Equipment-Showcase-Setup.md`. General action-bar integration does not require changing your Animation Blueprint.

Edit equipment-category icons

1. Open `BP_ABK_SimpleAbilityRouter` > Class Defaults. In Details search Weapon Category Icons; expand the array under Auto Attack > Equipment Icon.
2. Keep None for empty hands, then edit each supported category's texture reference. Add a row for another category your equipment system supports. The current eight categories are None, Sword, Axe, Mace, Bow, Dagger, Spear and Staff.
3. Call Set Equipped Weapon Category to inspect acceptance/Failure, or use the same-pawn equipment source's events. An unmapped category retains the last known icon. Category changes do not replace punches with weapon combat.
4. Compile/save and test equip, replace and unequip. Walkthrough 10 shows the actual expanded mapping. All eight native icon previews were saved and verified after a fresh editor reload.

Support and reference

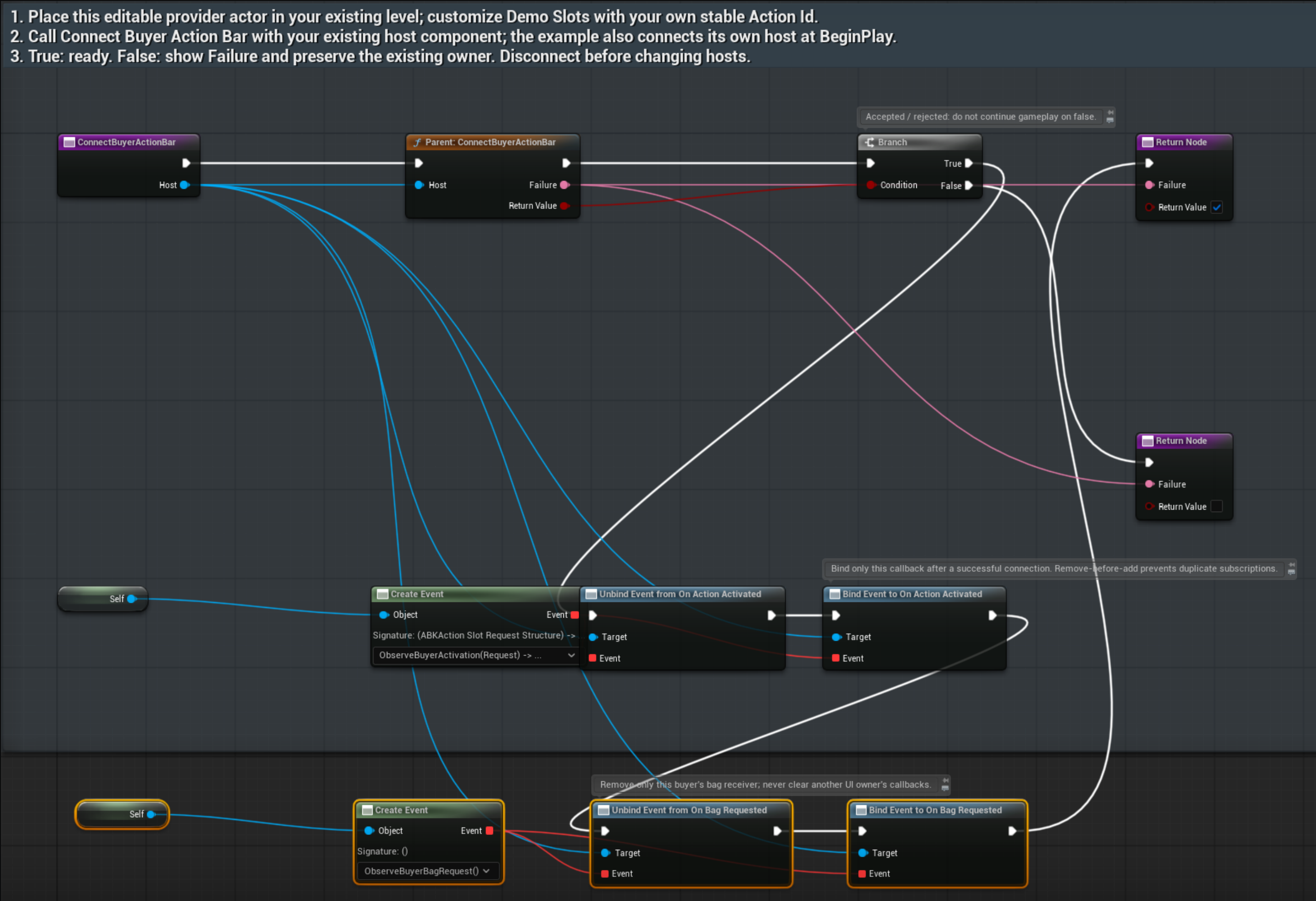
Contact support.blackironspine@proton.me. Target initial response is within 24 hours; this is not a promised resolution time.

[Detailed settings reference](#) | [Blueprint quick start](#) | [Equivalent native integration](#) | [Full guide \(PDF\)](#) | [Optional equipment/showcase setup](#)

Blueprints: connect your game

Connect your gameplay owner

/ActionBarKit/ActionBarKit/Examples/BP_ABK_BuyerProvider Graph / function: ConnectBuyerActionBar



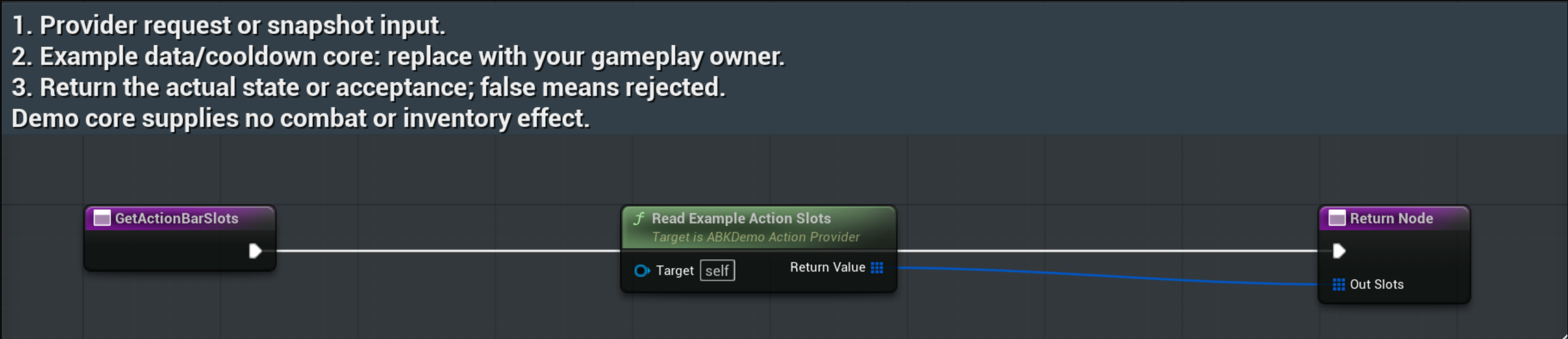
What to connect and what happens

1. Duplicate the supplied example into your project if you want to customize it. Place your provider actor in the level and configure Demo Slots with your own stable ActionId values.
2. Call Connect Buyer Action Bar with the existing Action Bar Host. Follow the Boolean branch: true connects the observer events; false returns Failure and preserves the existing owner.
3. Bind only after connection succeeds. The example removes each exact receiver before binding it, preventing duplicate action and bag callbacks. Compile, save and confirm your own callback runs once.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Supply current action data

/ActionBarKit/ActionBarKit/Examples/BP_ABK_BuyerProvider Graph / function: GetActionBarSlots



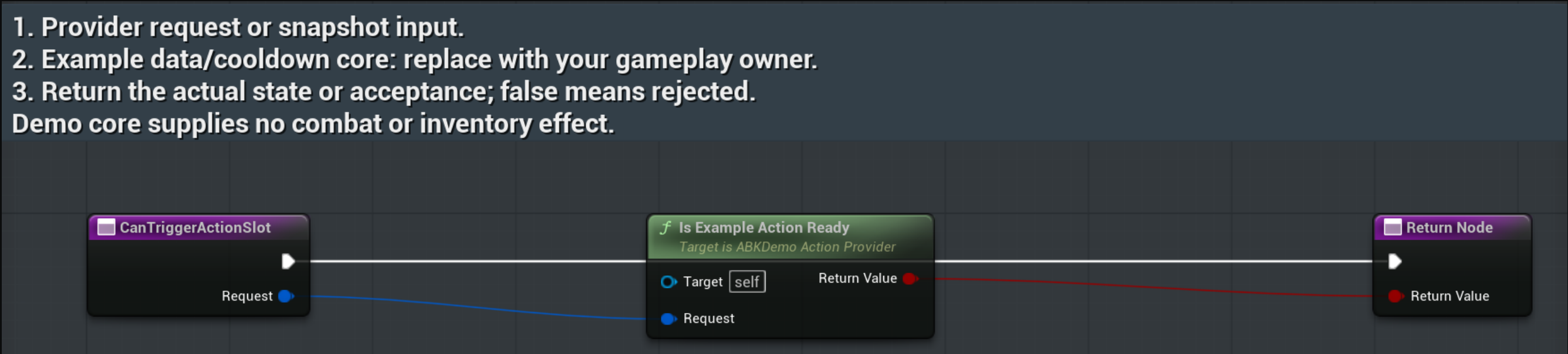
What to connect and what happens

1. Open GetActionBarSlots. The sample Read Example Action Slots feeds Out Slots; replace that snapshot source with data from your own gameplay owner.
2. Return unique positive positions, stable IDs, icons, complete key chords, availability, cooldowns and counts. Keep assignment/resource authority in your existing system.
3. Compile and save. Refresh From Action Provider should show your returned snapshot; an empty snapshot does not supply playable actions.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Decide whether an action is ready

/ActionBarKit/ActionBarKit/Examples/BP_ABK_BuyerProvider Graph / function: CanTriggerActionSlot



What to connect and what happens

1. Open CanTriggerActionSlot and pass Request to your readiness check. The sample Is Example Action Ready returns the Boolean shown.
2. Check current ID/position, cooldown, resources and permission without committing an effect. Return false for refusal.
3. Commit once in TriggerActionSlot after a final state check, then return true. Walkthrough 09 shows the supplied commit graph; this readiness function must stay free of gameplay effects.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Try request, cancel and reconnect

/ActionBarKit/ActionBarKit/Examples/BP_ABK_BuyerProvider Graph / function: BuyerInputAndCleanup



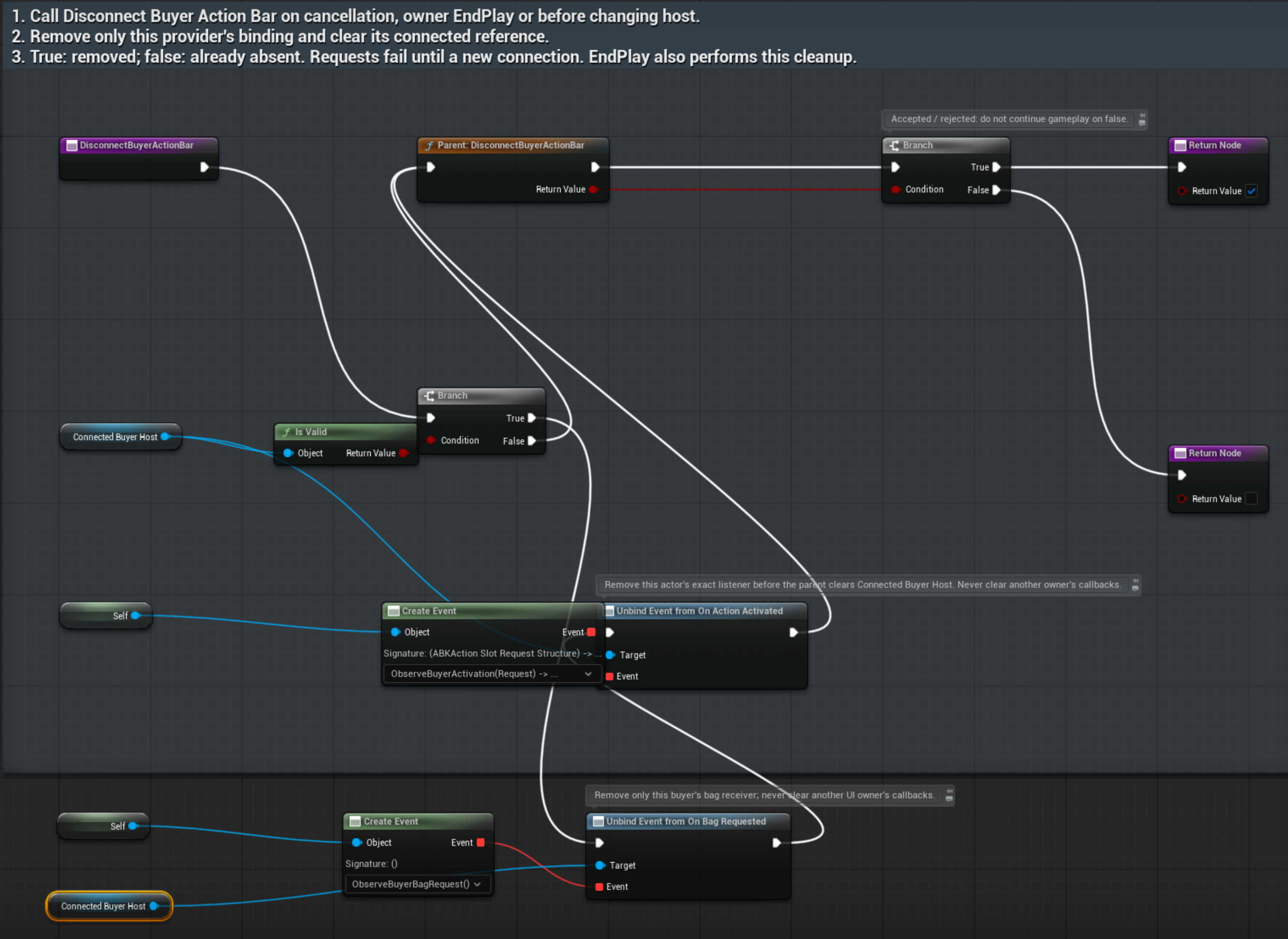
What to connect and what happens

1. Open BuyerInputAndCleanup on the supplied example actor. F8, F9 and F10 are actual sample key events in that actor: F8 requests slot 1, F9 disconnects and F10 reconnects. They are not global action-bar shortcut assignments.
2. Replace F8 with your game's input event. Store the returned acceptance and Failure, then handle the separate accepted/refused outcomes. Let the provider own the effect.
3. Test F9 followed by F8: the disconnected request must be refused. F10 reconnects this same actor's host. Disconnect first before changing hosts; another owner must remain protected.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Disconnect without clearing another owner

/ActionBarKit/ActionBarKit/Examples/BP_ABK_BuyerProvider Graph / function: DisconnectBuyerActionBar



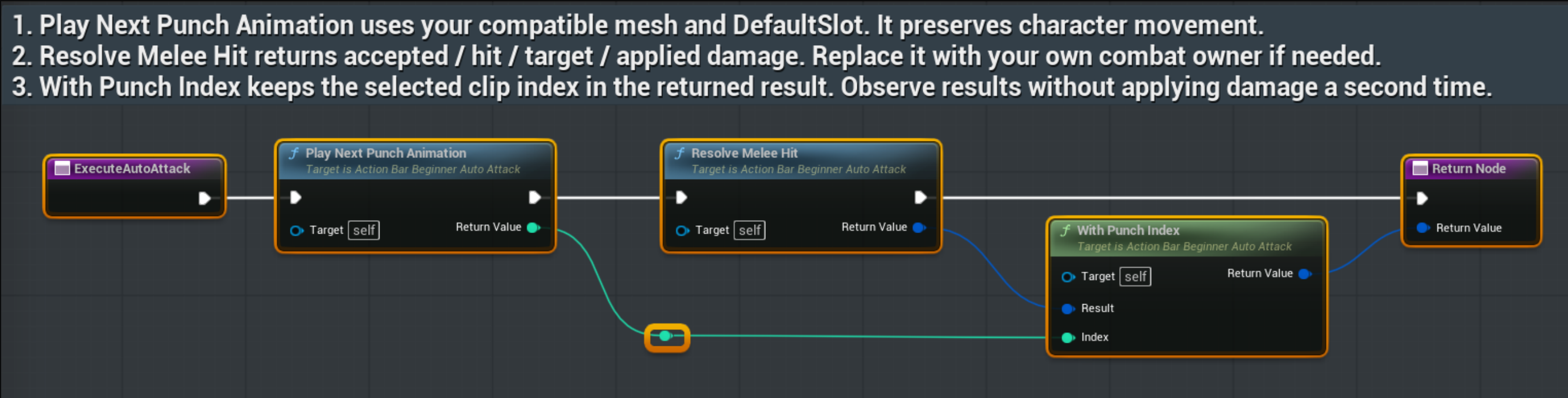
What to connect and what happens

1. Call Disconnect Buyer Action Bar on cancellation, owner EndPlay or before changing hosts.
2. Remove only this provider's exact action/bag event receivers before the parent clears its connected host reference.
3. True means removed; false means already absent. Requests fail until a valid new connection. Compile/save, disconnect twice and verify no stale callback fires.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Connect your own combat result

/ActionBarKit/ActionBarKit/Beginner/BP_ABK_SimpleAbilityRouter Graph / function: ExecuteAutoAttack



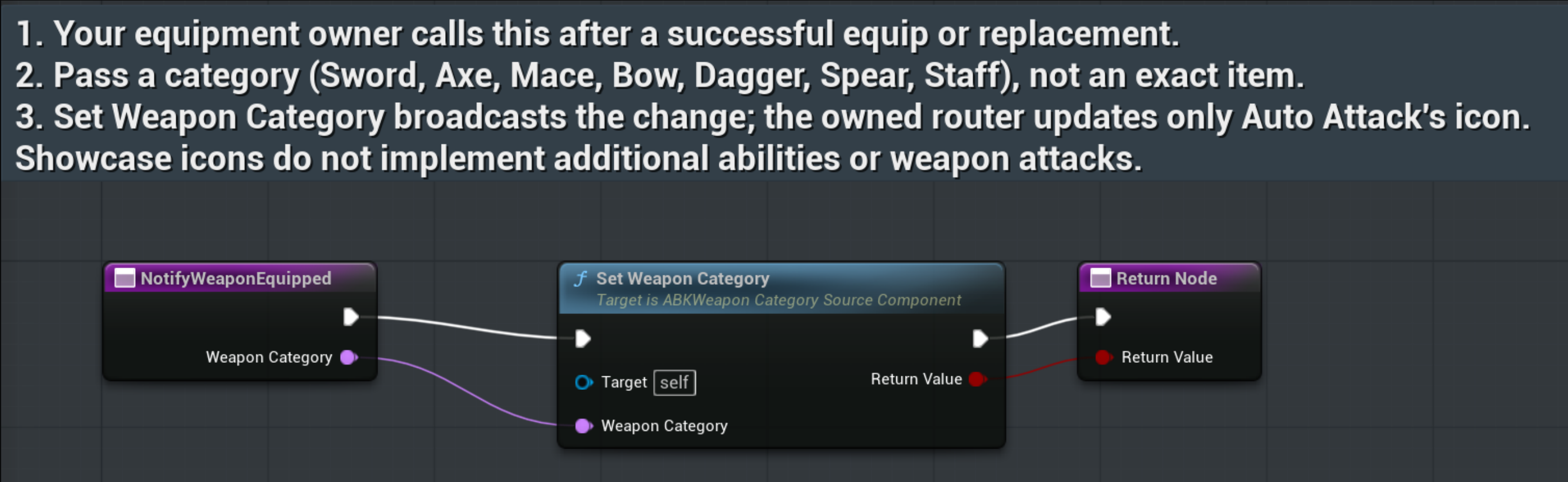
What to connect and what happens

1. Open ExecuteAutoAttack. Play Next Punch Animation cycles compatible clips on Animation Slot; Resolve Melee Hit makes the immediate sweep/damage request.
2. With Punch Index preserves the selected clip in the returned result. A miss can still be accepted; cooldown or authority refusal must not apply another effect.
3. For your combat system, replace the effect with your owner's call and return its actual result. Avoid both the parent damage effect and your replacement. Observe On Auto Attack Completed or Last Result.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Update the icon after equip or replacement

/ActionBarKit/ActionBarKit/Beginner/Examples/BP_ABK_EquipmentIconSource Graph / function: NotifyWeaponEquipped



What to connect and what happens

1. Attach the equipment source and your router to the same pawn. Your equipment owner calls Notify Weapon Equipped after it successfully equips or replaces the active weapon.
2. Pass Sword, Axe, Mace, Bow, Dagger, Spear or Staff through Weapon Category. The category represents the type of weapon; it need not identify the exact item.
3. Set Weapon Category broadcasts a change. Check the result and icon while preserving cooldown and other slots. An unsupported category is refused; your owner still owns the item and equip rules.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Restore the fist after unequip

/ActionBarKit/ActionBarKit/Beginner/Examples/BP_ABK_EquipmentIconSource Graph / function: NotifyWeaponUnequipped



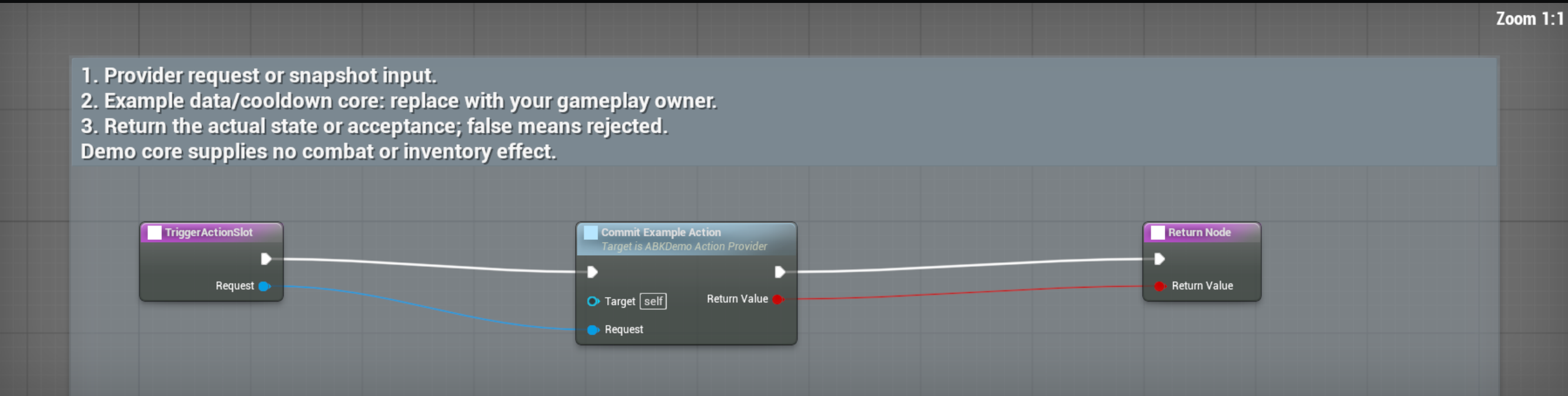
What to connect and what happens

1. When the active hands become empty, your equipment owner calls Notify Weapon Unequipped on that same pawn's source.
2. The graph sets category None, selecting the BIS fist. Fist and Unarmed names also normalize to None.
3. Compile/save and test equip, replacement and unequip. Other slots and cooldowns should be unchanged. Repeating the same category does not broadcast another change.

Capture status: Intact user-supplied graph capture. The matching Blueprint asset is reused in the validated release candidate.

Return your game's accepted action result

/ActionBarKit/ActionBarKit/Examples/BP_ABK_BuyerProvider - TriggerActionSlot



What to connect and what happens

1. Open BP_ABK_BuyerProvider and search TriggerActionSlot. This editable example returns the example provider's actual acceptance result.
2. Replace the example core with your gameplay owner's request. Return true only when that owner accepts the action; return false when it refuses.
3. Test an accepted request and a refused request. Observe the outgoing action event without applying the same gameplay effect twice. Keep cancellation and ownership cleanup connected.

Actual native capture of the validated candidate or identified compatible host animation example. The image remains whole.

Edit the weapon-category icon mapping

/ActionBarKit/ActionBarKit/Beginner/BP_ABK_SimpleAbilityRouter - Class Defaults - Weapon Category Icons

Weapon Category Icons				
Auto Attack				
Equipment Icon				
Weapon Category Icons			8 Array elements	 
Index [0]			2 members	
Weapon Category			None	
Icon				T_ABK_AutoAttackPunch   
Index [1]			2 members	
Weapon Category			Sword	
Icon				T_ABK_AutoAttackSword   
Index [2]			2 members	
Weapon Category			Axe	
Icon				T_ABK_Starter_EMBER   
Index [3]			2 members	
Weapon Category			Mace	
Icon				T_ABK_AutoAttackMace   
Index [4]			2 members	
Weapon Category			Bow	
Icon				T_ABK_AutoAttackBow   
Index [5]			2 members	
Weapon Category			Dagger	
Icon				T_ABK_AutoAttackDagger   
Index [6]			2 members	
Weapon Category			Spear	
Icon				T_ABK_AutoAttackSpear   
Index [7]			2 members	
Weapon Category			Staff	
Icon				T_ABK_AutoAttackStaff   

What to connect and what happens

1. Enable Show Plugin Content. Open BP_ABK_SimpleAbilityRouter, select Class Defaults, and search Weapon Category Icons in Details. Expand all eight array entries.
2. Each entry pairs a weapon category with a texture. None supplies the empty-hand BIS fist; Sword, Axe, Mace, Bow, Dagger, Spear and Staff supply category icons. Choose a category icon rather than an exact item image.
3. Compile and save. Call the equipment source's Notify Weapon Equipped after a successful equip/replacement and Notify Weapon Unequipped when hands become empty. Check that only Auto Attack's icon changes and cooldowns remain intact.

Actual native capture of the validated candidate or identified compatible host animation example. The image remains whole.