RED HANDED ACTIONS

RedHanded Actions define interactive behaviors that may trigger RedHanded repercussions when performed by an actor. Each action is created through a RedHandedActionSO, allowing designers to configure exactly how the action behaves—whether it is instant or lasting, abortable or non-abortable, harmless or punishable, and whether repercussions occur automatically or only when the actor is detected.

The system does not impose predefined action categories. Instead, users design their own action logic using the parameters provided by the RedHandedActionSO.

The demo scene includes several example patterns to illustrate common use cases, but developers are free to create any action style their game requires.

Before You Begin – About RedHanded Actions

RedHanded Actions are an **advanced feature** of the RedHanded System.

They are best approached **after** understanding **RedHanded Zones** and **RedHanded Objects**, as Actions combine both systems into interactive, player-driven behaviors.

Unlike Zones, which apply repercussions based purely on location, RedHanded Actions require a **small amount of coding**—typically one or two method calls placed inside your interaction logic.

This provides precise control over how and when actions are triggered, while keeping the API minimal and user-friendly.

Repercussions

A **Repercussion** is the consequence applied to an actor when they are caught performing a RedHanded Action.

In most cases, this consequence is a change of the actor's **Faction Name**, signaling that the actor has committed an unauthorized or criminal behavior.

A repercussion is applied **only if** the actor's current Faction Name appears in the action's **Required Factions** list.

When the punishment conditions for the action are met, the actor's Faction Name is reassigned to the **Result Faction**—indicating that the actor has been "caught red-handed."

Unlike RedHanded Zones, which rely on four predefined Zone Modes, **RedHanded Actions determine when repercussions occur based entirely on the action's configuration.**

Designers can create actions that are:

- **Immediate** – repercussions occur the moment the action is performed
- **Conditional** – repercussions occur only if the actor is detected
- **Completion-based** – repercussions occur only when the action fully completes
- **Avoidable** – repercussions occur only if the actor does not abort the action
- **Unavoidable** – repercussions occur once the action begins and cannot be interrupted

This flexible design allows a wide range of gameplay behaviors to be built from the same system.

Example Action Patterns (Demo Scene)

The demo scene showcases several common patterns, presented from simplest to most advanced to support a smooth learning curve:

- Pickpocketing - Instant Action - An instant action that triggers repercussions only when the actor is detected during the moment of the action.
- Wire Cutting - Lasting, Abortable Action - A hold-to-perform action that can be aborted at any time. Repercussions apply only if the actor is detected during the cutting process.
- Generator Sabotage - Completion-Based Lasting Action -A long action that must reach full completion. Repercussions are applied once the action is fully performed, and may also trigger **global consequences** within the environment.

These examples demonstrate how RedHanded Actions can represent anything from minor interactions to high-risk criminal activities- each with its own rules determining when the actor should be punished.

Example 1 (Action Instant: Pickpocketing)

- Actor Faction Name= Prisoner
- Required Faction Names= Prisoner
- Result Faction Name= Runner
- Apply If Got Caught = True

Result: Pickpocketing appears instantaneous to the player, but internally the system opens a very short action window to evaluate detection.

If the actor is detected at any point during this internal window—even by a minimal awareness delta—the repercussion is applied and the actor's Faction Name changes to Runner.

If no detection occurs during this brief moment, the action completes with no repercussions.

Example 1 - How the Action Works

1. Player presses the interaction button

    - The action is executed immediately.
    - There is no progress bar and no holding requirement.
    - The system briefly considers the player to be "performing an instant action."

2. The instant action completes

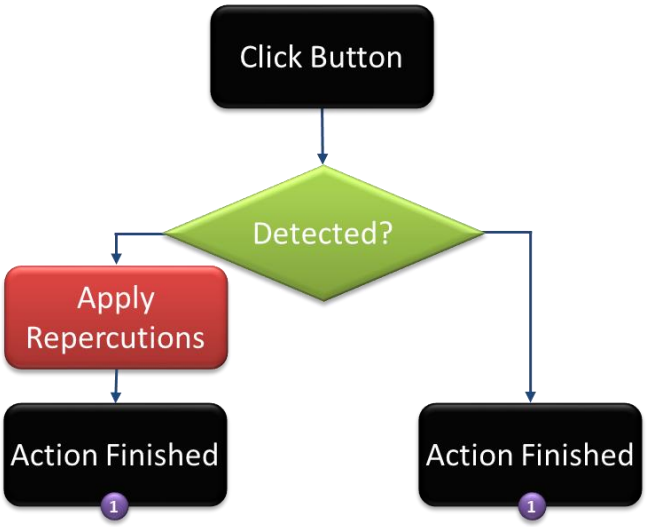
    - The system calls FinishInstantAction() to finalize the action.
    - RedHanded Component evaluates whether the action should trigger repercussions.
    - Since Apply If Got Caught is true, punishment happens only if the player was detected during this short action moment.

3. If the player was not detected

    - The action completes with no repercussions.
    - The stolen item is added to the player's inventory or state.

This makes instant actions feel immediate to the player while still allowing RedHanded to determine whether the act was witnessed.

Example 1 – Flow Diagram – With Code Cast References



1 FinishInstantAction(RedHandedActionSO);

Called once when the player conducts an instant action.

This informs the RedHanded Component to start monitoring detection event after action.

Example 2 (Action Lasting: Wire cutting)

- Actor Faction Name= Prisoner
- Required Faction Names= Prisoner
- Result Faction Name= Runner
- Apply If Got Caught = True

Result: Wire cutting is a lasting, abortable action. The player must hold the interaction button long enough to fill the progress bar and complete the action. While the action is in progress, the system tracks whether the player is detected—but repercussions are applied only if the action successfully finishes.

If the player aborts the action before completion, the progress resets and no repercussions occur, even if the player was detected during the attempt..

Example 2 - How the Action Works

1. Player holds the interaction button

    - The action begins.
    - A progress bar starts filling over time.
    - The system considers the player to be "performing a lasting action."

2. If the player keeps holding long enough to reach 100%

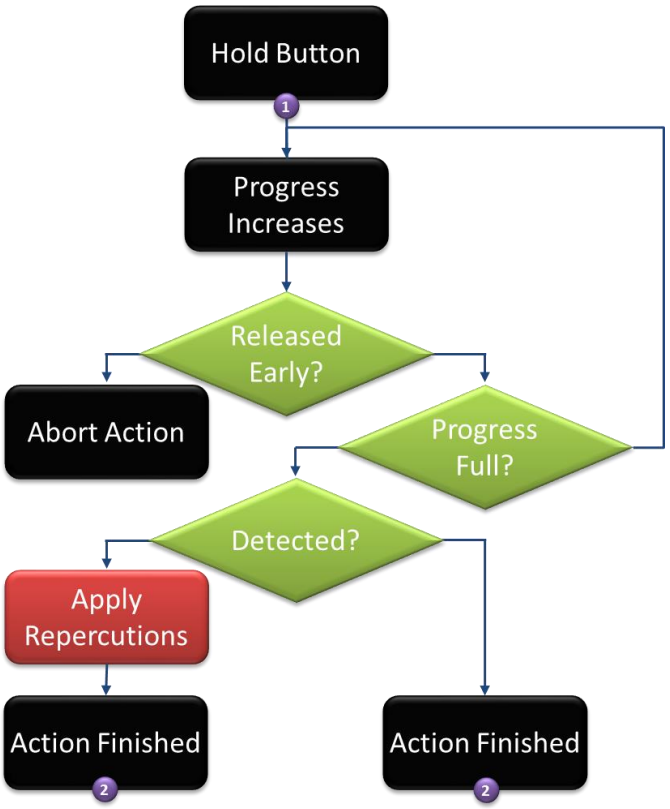
    - The action is completed successfully.
    - RedHanded checks whether the action should trigger repercussions.
    - Since Apply If Got Caught is true, punishment happens only if the player was detected at any point during the process.

3. If the player releases the button early

    - The action is canceled.
    - Progress resets to 0%.
    - No repercussions are applied.

This gives players time to stop an illegal action if they notice a guard approaching.

Example 2 – Flow Diagram – With Code Cast References



1 StartLastingAction()

Called once when the player begins a lasting action.

This informs the RedHanded Component to start monitoring detection during the action.

2 FinishLastingAction(RedHandedActionSO)

Called only when the action successfully reaches full progress.

The RedHanded Component uses the provided ActionSO to determine whether repercussions should be applied.

Example 3 (Generator sabotage)

- Actor Faction Name= Prisoner
- Required Faction Names= Prisoner
- Result Faction Name= Runner
- Apply If Got Caught = True
- Apply Globally = True

Result: Generator sabotage is a lasting, non-abortable action that must reach full completion before any consequences are evaluated. While the action is in progress, the system tracks whether the player is detected—but repercussions are applied only if the action successfully finishes.

If the action reaches completion, the system checks whether the player was detected during any part of the sabotage. If detection occurred, the player receives the defined repercussion, and the action may also trigger global repercussions, affecting every actor in the scene whose Faction Name matches the action's criteria.

If the player interrupts or fails to complete the sabotage (for example, releasing the interaction button or being forced to stop), the progress resets and no repercussions occur, even if the player was detected during the attempt.

Example 3 - How the Action Works

1. Player holds the interaction button

    - The action begins.
    - A progress bar starts filling over time.
    - The system considers the player to be "performing a lasting action."

2. If the player keeps holding long enough to reach 100%

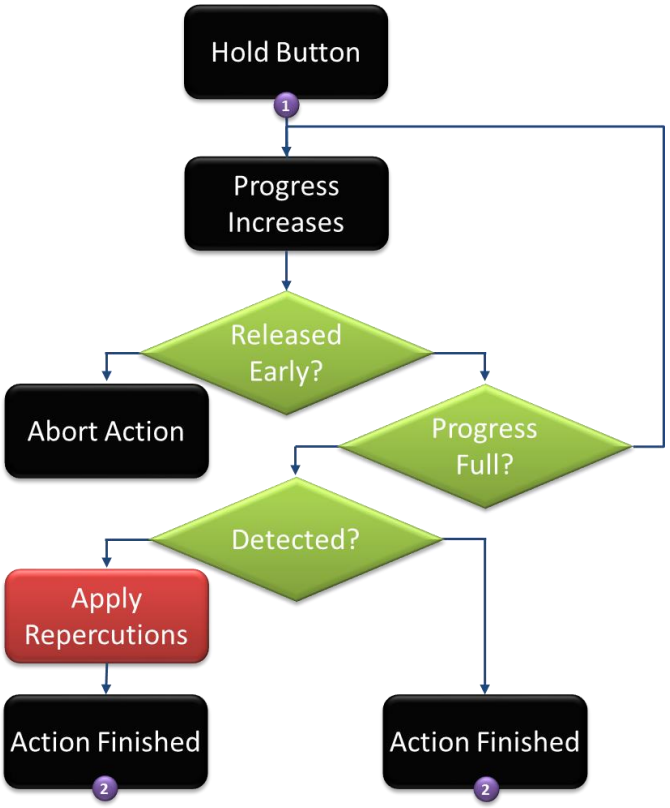
    - The action is completed successfully.
    - RedHanded checks whether the action should trigger repercussions.
    - Since Apply If Got Caught is true, punishment happens only if the player was detected at any point during the process.
    - Since Apply Globally is true and a repercussion was applied, RedHanded then checks all RedHanded Components in the scene to determine which actors also qualify for global repercussions.

3. If the player releases the button early

    - The action is canceled.
    - Progress resets to 0%.
    - No repercussions are applied.

This gives players time to stop an illegal action if they notice a guard approaching.

Example 3 – Flow Diagram – With Code Cast References



1 StartLastingAction()

Called once when the player begins a lasting action.

This informs the RedHanded Component to start monitoring detection during the action.

2 FinishLastingAction(RedHandedActionSO)

Called only when the action successfully reaches full progress.

The RedHanded Component uses the provided ActionSO to determine whether repercussions should be applied, and applied them globally as needed in example 3.

RedHanded Action SO

RedHandedActionSO is a ScriptableObject asset that defines the configuration of a RedHanded Action.

Each ActionSO describes what kind of action the player is performing and under what conditions this action should trigger repercussions.

This keeps the action logic clean, reusable, and fully data-driven, allowing designers to adjust behavior without modifying any code.

1	Required Factions	The list of factions allowed to perform the action without instantly causing punishment.
2	Result Faction	The faction assigned to the actor if repercussions are triggered.
3	Apply If Got Caught	Determines whether the action is only punishable when the actor is detected.
4	Apply Globally	Determines whether repercussions should propagate to other actors with matching faction rules.

Creating New RedHanded Action SO

To create a new RedHanded Action asset, open the Unity Project Window, right-click inside any folder, and choose:

Create → CatBorg Studio → Senses → RedHandedActionSO.

This generates a new ScriptableObject that you can configure directly in the Inspector—select the Required Factions, Result Faction, and choose whether the action should apply repercussions only when detected or globally. Once created, the ActionSO can be assigned to any interaction script or system that triggers RedHanded Actions.