

# ItemOwnership

V1.0.57 by Lurch

A utility which prevents **special** items from being used by other players unless the items are in the shareable items list.

## **Important Notes:**

1. This plugin is NOT a standalone plugin. It is designed to work with other plugins on modded Rust servers. You must either modify an existing plugin, have the author of an existing plugin modify theirs to work with ItemOwnership, or have a plugin that already works with ItemOwnership to get the benefit of this plugin.
2. **This plugin has not been tested with items that are stackable.** We expect that the underlying FacePunch code will utterly break the original item's UID, so there's no support at this time for stacked special items.

It is up to you, the developer / system op to ensure that special items in your server that you wish to prevent sharing between players cannot be stacked.

## **Operation:**

ItemOwnership ( IO ) keeps track of the Unique Item Id's for specially crafted / given items.

For example, a special player-usable Invisibility Cloak (available: [MyVector.xyz](https://myvector.xyz) or [CodeFling.com](https://codefling.com)) that works directly with IO. IO will also work with any other plugin that uses the IO API.

The API can be called by an by an in-game /kit command or from an NPC vendor.

When your special or unique item is created, the plugin responsible for the item creation must call the IO API hook, **UpdateItemList()** – this hook allows adding new items, changing ownership of existing items and deleting items from the registry. (Usually called when an item is destroyed.)

From the time the new item is registered to an owner, any other player will be unable to:

- Pick up the item.
- Equip or use the item.
- Transfer the item to inventory.
- Move the item in any inventory.

## **Shareable Items:**

You can place any regular Rust item ID into the Configuration class under `ShareableItemIds`, or “List of Rust Item ID's that are shareable” in the Configuration file.

ItemOwnership ( IO ) comes with two shareable items preconfigured – the Chainsaw and Jackhammer.

A good example for sharing these two items is on Dr. Evil's Zombiedaddon: the “Chippy” chainsaw and “Jack-o-hammer” jackhammer are very expensive items to craft as they are also powerful tools.

These items are set to shared so that while most items can't be passed down (funneled) to lower-tier players, the chainsaw and jackhammer *can* be shared with other players, like in a shared-base or team setting.

To add a shareable item, just add the item's Rust ID to the list mentioned above. You can find and lookup any Rust Item ID by following this link:

<https://www.corrosionhour.com/rust-item-list/>

Just be sure to correctly format the JSON field, and separate new values with commas. (Leave the last value in the list without a comma!)

## **Permissions:**

ItemOwnership uses Oxide permissions.

ItemOwnership.admin  
ItemOwnership.bypass

These two perms are mutually exclusive. Admin permission does not give bypass, and bypass does not give admin permission.

**Admin** Allows you to use admin-only commands. (See next section; chat commands.)

**Bypass** allows usage of special/registered items without restriction, regardless of who owns them.

### **Chat commands:**

#### **/io.reload**

Admin permission required. Causes plugin to fetch everything from stored file. With debug mode on, this command will print a complete list of all items and owners to which they belong.

#### **/io.ver or /io.version**

No perms required. Prints the current plugin version.

#### **/io.changeowner or /io.co**

Admin permission required. Acts as a bulk transfer command – changes ALL owner ID's to the specified owner ID or zero (which means no owner and effectively prohibits player from using items.)

### **Syntax:**

`/io.co <FROM ID> <TO ID>` The TO ID is optional, and if left out, will result in all items assigned to the FROM ID to be assigned to player ID zero. (0).

e.g:

`/io.co 76561197970521596 Thagmeister`

The result of the above command is all special items belonging to Lurch (yours truly) will be changed to reflect ownership by 'Thagmeister' – a different player. You could also use:

`/io.co Lurch Thagmeister`, or `/io.co Lurch` <- which assigns everything Lurch owns to effectively no owner. Poor Lurch won't be able to use any of his special in game items at all. Special note here, by omitting the second argument, the effective owner becomes an invalid SteamID (but is still a Steam ID.)

You can reverse the operation without having to know the special Steam ID that the items now belong to, simply by using a '!' or '0' (zero) in the first argument, and the second argument would be the recipient's SteamID.

e.g. `/io.co ! Lurch`

#### **/io.debug**

Admin permission required. Toggles the plugin debug state to on or off.

## API:

Some things to keep in mind:

- When calling the IO hook to add items, be sure you're also calling the same hook to delete items when they are destroyed or removed in the game.
- Don't try to track mundane items – your IO database will fill up quickly and lookups might begin to slow down player's framerates. (We've seen it go over 5,000 entries w/o degradation, just try not to push it!)
- Again, don't expect IO to work with stackable items, as the UID's are lost and re-created when individual items are pulled back off the stack.

**Void UpdateItemList(BasePlayer owner, Item item, int action)**

Where:

**Owner** is a BasePlayer object (the item's intended owner)

**Item** is the actual in-game item

**Action** is one of:

- 0 = Add new
- 1 = Change Owner
- 2 = Delete (Item was destroyed)

**Bool? IsItemOwner(BasePlayer player, Item item)**

Returns null / true / false under these conditions:

**null** = Unable to find the item in the IO registry  
**true** = Specified player IS the item's owner  
**false** = Specified player is NOT the item's owner.

Example C# code to call into IO API to add item to tracking file:

```
Interface.CallHook("UpdateItemList", player, myItem, 0); // 0 = add
```

Good luck with ItemOwnership! It is free for public use – licensed under the MIT License model (see source file for details.)

## Support:

Limited support is available on Lurch's lounge: <https://discord.gg/kwjnmnSYps>